Amazon Relational Database Service

User Guide

API Version 2014-10-31

Amazon Relational Database Service: User Guide

Copyright © 2015 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Table of Contents

What Is Amazon RDS? ... 1
- Amazon RDS Components ... 2
  - DB Instances ... 2
  - Regions and Availability Zones ... 3
  - Security Groups ... 3
  - DB Parameter Groups ... 3
  - DB Option Groups ... 3
- Available RDS Interfaces ... 4
  - Amazon RDS Console ... 4
  - Command Line Interface ... 4
  - Programmatic Interfaces ... 4
- How You Are Charged for Amazon RDS ... 5
- Monitoring an Amazon RDS DB Instance ... 5
- What's Next? ... 5
  - Getting Started ... 5
  - Database Engine Specific Topics ... 6

Setting Up ... 7
- Sign Up for AWS ... 7
- Create an IAM User ... 8
- Determine Requirements ... 9
- Create a Security Group ... 10

Getting Started ... 12
- Creating a MySQL DB Instance and Connecting to a Database ... 12
  - Creating a MySQL DB Instance ... 13
  - Connecting to a Database on a DB Instance Running MySQL ... 17
  - Deleting a DB Instance ... 18
- Creating an Oracle DB Instance and Connecting to a Database ... 18
  - Creating a DB Instance Running Oracle ... 18
  - Connecting to a DB Instance Running Oracle ... 24
  - Deleting a DB Instance ... 26
- Creating a SQL Server DB Instance and Connecting to a Database ... 26
  - Creating a SQL Server DB Instance ... 26
  - Connecting to a SQL Server DB Instance Using SQL Server Management Studio ... 31
  - Troubleshooting a Connection to a DB Instance Running SQL Server ... 35
  - Deleting a DB Instance ... 36
- Creating a PostgreSQL DB Instance and Connecting to a Database ... 36
  - Creating a PostgreSQL DB Instance ... 36
  - Connecting to a PostgreSQL DB Instance ... 43
  - Deleting a DB Instance ... 46
- Creating a DB Cluster and Connecting to a Database on an Amazon Aurora DB Instance ... 47
  - Create a DB Cluster ... 47
  - Connect to an Instance in a DB Cluster ... 52
  - Delete the Sample DB Cluster, DB Subnet Group, and VPC ... 52

Best Practices ... 54
- Amazon RDS Basic Operational Guidelines ... 54
- DB Instance RAM Recommendations ... 55
- Amazon RDS Security Best Practices ... 55
- Using Metrics to Identify Performance Issues ... 55
  - Viewing Performance Metrics ... 56
  - Evaluating Performance Metrics ... 57
  - Tuning Queries ... 59
- Best Practices for Working with MySQL Storage Engines ... 59
- Best Practices for Working with PostgreSQL ... 60
  - Loading Data into a PostgreSQL DB Instance ... 60
  - Working with the fsync and full_page_writes database parameters ... 60

Working with the PostgreSQL Autovacuum Feature ... 60
Best Practices for Working with SQL Server ... 62
Amazon RDS SQL Server Best Practices Video ... 62
Amazon RDS Best Practices Presentation Video ... 63
DB Instances ... 64
DB Instance Class ... 65
DB Instance Status ... 68
Regions and Availability Zones ... 69
Related Topics ... 71
High Availability (Multi-AZ) ... 71
Failover Process for Amazon RDS ... 72
Amazon RDS and Amazon VPC ... 73
DB Instance Backups ... 73
Automated Backup ... 74
DB Snapshots ... 76
Related Topics ... 76
DB Instance Replication ... 76
Storage ... 77
Storage Types ... 77
Performance Metrics ... 78
Facts About Amazon RDS Storage ... 78
Other Factors That Impact Storage Performance ... 79
Adding Storage and Changing Storage Type ... 80
General Purpose (SSD) Storage ... 80
I/O Credits and Burst Performance ... 80
Provisioned IOPS Storage ... 82
Using Provisioned IOPS Storage with Multi-AZ, Read Replicas, Snapshots, VPC, and DB Instance Classes ... 83
Provisioned IOPS Storage Costs ... 83
Getting the Most out of Amazon RDS Provisioned IOPS ... 84
Provisioned IOPS Storage Support in the CLI and Amazon RDS API ... 84
Factors That Affect Realized IOPS Rates ... 85
Page Size and Channel Bandwidth ... 85
DB Instance Class ... 85
Database Workload ... 86
Security ... 87
Using IAM to Manage Access to Amazon RDS Resources ... 88
Creating IAM Policies for Amazon RDS ... 89
How Resource Authorization Works in Amazon RDS ... 90
Specifying Conditions in an IAM Policy for Amazon RDS ... 91
Example IAM Policies for Amazon RDS ... 93
Supported Resource-Level Permissions for Amazon RDS API Actions ... 96
Encrypting Amazon RDS Resources ... 106
Enabling Amazon RDS Encryption for a DB Instance ... 107
Availability of Amazon RDS Encrypted Instances ... 107
Managing Amazon RDS Encryption Keys ... 108
Limitations of Amazon RDS Encrypted Instances ... 108
Using SSL to Encrypt a Connection ... 109
SSL Certificate Rotation ... 109
Amazon RDS Security Groups ... 110
DB Security Groups ... 111
VPC Security Groups ... 111
DB Security Groups vs. VPC Security Groups ... 111
Security Group Scenario ... 112
Related Topics ... 113
Limits ... 114
Limits in Amazon RDS ... 114
Naming Constraints in Amazon RDS ... 115

File Size Limits in Amazon RDS ... 117
MySQL File Size Limits in Amazon RDS ... 117
MySQL on Amazon RDS ... 119
MySQL Planning Information ... 120
MySQL Versions ... 120
Amazon RDS Supported Storage Engines ... 121
Amazon RDS and MySQL Security ... 122
InnoDB Cache Warming ... 123
MySQL Features Not Supported By Amazon RDS ... 124
Known Issues and Limitations ... 125
Creating a DB Instance Running MySQL ... 129
AWS Management Console ... 205
CLI ... 135
API ... 136
Related Topics ... 136
Connecting to a DB Instance Running MySQL ... 137
Connecting from the MySQL Utility ... 137
Connecting with SSL ... 138
Maximum MySQL connections ... 138
Related Topics ... 139
Modifying a DB Instance Running MySQL ... 140
AWS Management Console ... 140
CLI ... 142
API ... 142
Importing and Exporting Data From a MySQL DB Instance ... 143
Overview ... 143
Importing Data Considerations ... 143
Importing Data from a MySQL DB to an Amazon RDS MySQL DB Instance ... 146
Importing Data to an Amazon RDS MySQL DB Instance with Reduced Downtime ... 147
Importing Data From Any Source to a MySQL DB Instance ... 159
Replication with a MySQL Instance Running External to Amazon RDS ... 162
Using Replication to Export MySQL 5.6 Data ... 164
Appendix: Common DBA Tasks for MySQL ... 168
Killing a Session or Query ... 168
Skipping the Current Replication Error ... 168
Working with InnoDB Tablespaces to Improve Crash Recovery Times ... 169
Managing the Global Status History ... 171
Appendix: Options for MySQL ... 173
MySQL 5.6 memcached Support ... 173
Appendix: MySQL on Amazon RDS SQL Reference ... 177
Overview ... 177
SQL reference conventions ... 178
mysql.rds_set_external_master ... 178
mysql.rds_reset_external_master ... 180
mysql.rds_start_replication ... 180
mysql.rds_stop_replication ... 181
mysql_rds_skip_repl_error ... 182
mysql.rds_next_master_log ... 182
mysql.rds_innodb_buffer_pool_dump_now ... 184
mysql.rds_innodb_buffer_pool_load_now ... 185
mysql.rds_innodb_buffer_pool_load_abort ... 185
mysql.rds_set_configuration ... 186
mysql.rds_show_configuration ... 186
mysql.rds_kill ... 187
mysql.rds_kill_query ... 188
mysql.rds_rotate_general_log ... 188
mysql.rds_rotate_slow_log ... 189
mysql.rds_enable_gsh_collector ... 190

mysql.rds_set_gsh_collector ... 190
mysql.rds_disable_gsh_collector ... 191
mysql.rds_collect_global_status_history ... 191
mysql.rds_enable_gsh_rotation ... 191
mysql.rds_set_gsh_rotation ... 192
mysql.rds_disable_gsh_rotation ... 192
mysql.rds_rotate_global_status_history ... 193
Oracle on Amazon RDS ... 194
Planning Your Amazon RDS Oracle DB Instance ... 195
Oracle Database Engine Options ... 195
Security ... 203
Oracle Version Management ... 203
Licensing ... 203
Using OEM, APEX, TDE, and other options ... 204
Creating a DB Instance Running Oracle ... 205
AWS Management Console ... 205
CLI ... 210
API ... 210
Related Topics ... 211
Connecting to a DB Instance Running Oracle ... 212
Console ... 212
CLI ... 214
Related Topics ... 214
Modifying a DB Instance Running Oracle ... 216
AWS Management Console ... 216
CLI ... 218
API ... 218
Importing Data Into Oracle on Amazon RDS ... 219
Oracle SQL Developer ... 219
Oracle Data Pump ... 219
Oracle Export/Import Utilities ... 223
Oracle SQL*Loader ... 223
Oracle Materialized Views ... 224
Appendix: Options for Oracle ... 226
Oracle 11g Enterprise Manager (OEM) Database Control and Oracle 12c OEM Database Express ... 226
Oracle XML DB ... 227
Oracle Application Express (APEX) ... 227
Oracle Native Network Encryption ... 233
Oracle Transparent Data Encryption (TDE) ... 235
Oracle Statspack ... 236
Oracle Time Zone ... 239
Appendix: Common DBA Tasks for Oracle ... 241
Enabling and disabling Restricted Session ... 242
Flushing the Shared Pool ... 242
Flushing the Buffer Cache ... 242
Disconnecting a Session (for version 11.2.0.3.v1 and later) ... 243
Killing a Session ... 243
Renaming the Global Name (for version 11.2.0.3.v1 and later) ... 243
Granting Privileges to Non-Master Users ... 244
Modifying DBMS_SCHEDULER Jobs ... 244
Switching Online Log files ... 245
Adding, Dropping and Resizing Online Redo Logs ... 245
Setting Force Logging (for version 11.2.0.3.v1 and later) ... 248
Retaining Archived Redo Logs (for version 11.2.0.2.v7 and later) ... 248
Setting Supplemental Logging (for version 11.2.0.3.v1 and later) ... 248
Creating and Resizing Tablespaces and Data Files ... 249
Setting Default Tablespace ... 249

Setting Default Temporary Tablespace 249
Checkpointing the Database 250
Setting Distributed Recovery (for version 11.2.0.3.v1 and later) 250
Granting SELECT or EXECUTE privileges to SYS Objects (for version 11.2.0.3.v1 and later) 250
Setting the Database Time Zone 251
Working with Automatic Workload Repository (AWR) 251
Adjusting Database Links for Use with DB Instances in a VPC 251
Creating New Directories in the Main Data Storage Space (for version 11.2.0.4.v1 and later) 251
Listing and Reading Files in a DB Instance Directory (for version 11.2.0.3.v1 and later) 252
Appendix: Using Oracle GoldenGate with Amazon RDS 253
Setting Up an Oracle GoldenGate Hub on EC2 256
Setting Up a Source Database for Use with GoldenGate on Amazon RDS 258
Setting Up a Target Database for Use with GoldenGate on Amazon RDS 262
Working with Oracle GoldenGate's Extract and Replicat Utilities 264
Troubleshooting Issues When Using Oracle GoldenGate with Amazon RDS 267
Appendix: Using AWS CloudHSM to Store Amazon RDS Oracle TDE Keys 269
Setting Up AWS CloudHSM to Work with Amazon RDS 270
Setting Up Amazon RDS to Work with AWS CloudHSM 274
Verifying the HSM Connection, the Oracle Keys in the HSM, and the TDE Key 280
Restoring Encrypted DB Instances 282
Managing a Multi-AZ Failover 283
Appendix: Oracle Character Sets Supported in Amazon RDS 284
Appendix: Oracle Database Engine Release Notes 286
Database Engine Version: 11.2.0.2.v3 287
Database Engine Version: 11.2.0.2.v4 or 11.2.0.2.v5 287
Database Engine Version: 11.2.0.2.v6 288
Database Engine Version: 11.2.0.2.v7 289
Database Engine Version: 11.2.0.3.v1 290
Database Engine Version: 11.2.0.3.v2 291
Database Engine Version: 11.2.0.3.v3 293
Database Engine Version: 11.2.0.4.v1 294
Database Engine Version: 11.2.0.4.v2 (Deprecated) 295
Database Engine Version: 11.2.0.4.v3 295
Database Engine Version: 11.2.0.4.v4 296
Database Engine Version: 12.1.0.1.v1 297
Database Engine Version: 12.1.0.1.v2 298
Database Engine Version: 12.1.0.2.v1 299
Microsoft SQL Server on Amazon RDS 301
Common Management Tasks for SQL Server on Amazon RDS 301
Planning Your SQL Server DB Instance on Amazon RDS 302
General Limits for SQL Server DB Instances 303
Support for SQL Server Features on Amazon RDS 359
SQL Server Licensing 306
Planning your Multi-AZ Deployments Using SQL Server Mirroring 307
Database Engine Version Management 309
Supported SQL Server Roles and Permissions 310
Using SSL with a SQL Server DB Instance 310
Using the TDE Option to Encrypt Data at Rest 312
Creating a DB Instance Running SQL Server 313
AWS Management Console 313
CLI 319
API 319
Related Topics 320
Connecting to a DB Instance Running SQL Server 321
Connecting with SQL Server Management Studio 321
Connecting with SQL Workbench/J 324

Troubleshooting a Connection to a DB Instance Running SQL Server ... 326
Related Topics ... 326
Modifying a DB Instance Running SQL Server ... 328
AWS Management Console ... 328
CLI ... 330
API ... 330
Working with SQL Server Multi-AZ with Mirroring ... 331
Determining the Location of the Standby Mirror ... 331
Related Topics ... 331
Importing and Exporting SQL Server Data ... 332
Importing Data into SQL Server on Amazon RDS ... 332
Exporting Data from SQL Server on Amazon RDS ... 339
Appendix: Common DBA Tasks for SQL Server ... 342
Determining a Recovery Model ... 342
Collations and Character Sets for SQL Server ... 342
Resetting the db_owner Role Password ... 343
Transitioning a Database from OFFLINE to ONLINE ... 343
Dropping a Database in a Multi-AZ Deployment Using Mirroring ... 343
Analyzing Your Database Workload on a DB Instance Using SQL Server Tuning Advisor ... 344
Using SQL Server Agent ... 346
Working with SQL Server Logs ... 348
Handling UTC Times for Time Zone Awareness ... 348
Appendix: Options for SQL Server ... 350
SQL Server Transparent Data Encryption ... 350
Multi-AZ Deployment for SQL Server Using the Mirroring Option ... 353
PostgreSQL on Amazon RDS ... 355
Amazon RDS PostgreSQL Planning Information ... 356
Supported PostgreSQL Database Versions ... 357
Database Engine Features ... 359
Limits for PostgreSQL DB Instances ... 361
Minor Version Upgrades ... 361
Using SSL with a PostgreSQL DB Instance ... 361
Creating a DB Instance Running PostgreSQL ... 363
AWS Management Console ... 363
CLI ... 368
API ... 368
Related Topics ... 369
Connecting to a DB Instance Running the PostgreSQL Database Engine ... 370
Using pgAdmin to Connect to a PostgreSQL DB Instance ... 370
Using *psql* to Connect to a PostgreSQL DB Instance ... 372
Troubleshooting Connection Issues ... 372
Related Topics ... 373
Modifying a DB Instance Running PostgreSQL ... 374
AWS Management Console ... 374
CLI ... 376
API ... 376
Importing Data into PostgreSQL on Amazon RDS ... 377
Importing a PostgreSQL Database from an Amazon EC2 Instance ... 377
Using the \copy Command to Import Data to a Table on a PostgreSQL DB Instance ... 378
Appendix: Common DBA Tasks for PostgreSQL ... 380
Creating Roles ... 380
Managing PostgreSQL Database Access ... 380
Working with PostgreSQL Parameters ... 381
Setting up PostGIS ... 389
Using pgBadger for Log Analysis with PostgreSQL ... 391
Aurora on Amazon RDS ... 392
Availability ... 393
Aurora Endpoints ... 393

Amazon Aurora Storage ... 394
Amazon Aurora Replication ... 394
Amazon Aurora Reliability ... 394
Storage Auto-Repair ... 394
"Survivable" Cache Warming ... 395
Crash Recovery ... 395
Amazon RDS for Aurora Security ... 395
Securing Aurora Data with SSL ... 396
Using the memcached Option with Amazon Aurora ... 397
Comparison of Amazon RDS for Aurora and Amazon RDS for MySQL ... 397
Creating an Amazon Aurora DB Cluster ... 398
DB Cluster Prerequisites ... 399
Using the AWS Management Console to Launch an Aurora DB Cluster and Create an Aurora Replica ... 400
Creating a VPC for Aurora ... 407
Connecting to an Amazon Aurora DB Cluster ... 413
Connecting with SSL ... 414
Troubleshooting Aurora Connection Failures ... 415
Migrating Data to an Amazon Aurora DB Cluster ... 415
Migrating an RDS MySQL Snapshot to Aurora ... 415
Replication with Amazon Aurora ... 422
Monitoring Aurora Replication ... 423
Replication with MySQL ... 423
Monitoring an Amazon Aurora DB Cluster ... 425
Aurora Metrics ... 426
Managing an Amazon Aurora DB Cluster ... 428
Managing Performance and Scaling for Aurora DB Cluster ... 428
Backing Up and Restoring an Aurora DB Cluster ... 429
Fault Tolerance for an Aurora DB Cluster ... 430
Testing Amazon Aurora Using Fault Injection Queries ... 431
Best Practices with Amazon Aurora ... 433
Determining Which DB Instance You Are Connected To ... 434
Using Amazon Aurora to Scale Reads for Your MySQL Database ... 434
Using Amazon Aurora for Disaster Recovery with Your MySQL Databases ... 436
Migrating from MySQL to Amazon Aurora with Reduced Downtime ... 437
Appendix: DB Cluster and DB Instance Parameters ... 437
Cluster-level parameters ... 437
DB Instance Lifecycle ... 440
Upgrading and Amazon RDS Resource Maintenance ... 442
Operating System Updates for a DB Instance ... 443
Upgrading Database Versions for a DB Instance ... 446
Amazon RDS Maintenance Window ... 451
Modifying a DB Instance and Using the Apply Immediately Parameter ... 455
Renaming a DB Instance ... 457
AWS Management Console ... 457
CLI ... 458
API ... 458
Related Topics ... 458
Deleting a DB Instance ... 459
Deleting a DB Instance with No Final Snapshot ... 459
Deleting a DB Instance with a Final Snapshot ... 460
Related Topics ... 461
Rebooting a DB Instance ... 462
AWS Management Console ... 462
CLI ... 462
API ... 463
Working with Storage Types ... 464
Modifying a DB Instance to Use a Different Storage Type ... 464

Modifying IOPS and Storage Settings for a DB Instance That Uses Provisioned IOPS 466
Creating a DB Instance that Uses Provisioned IOPS Storage 468
Creating a MySQL Read Replica That Uses Provisioned IOPS Storage 470
Working with PostgreSQL and MySQL Read Replicas 472
Amazon RDS Read Replica Overview 472
PostgreSQL Read Replicas (version 9.3.5 and later) 473
MySQL Read Replicas 475
Creating a Read Replica 476
Promoting a Read Replica to Be a DB Instance 477
Replicating a Read Replica Across Regions (MySQL only) 479
Monitoring Read Replication 481
Troubleshooting a MySQL Read Replica Problem 483
Troubleshooting a PostgreSQL Read Replica Problem 484
Tagging Amazon RDS Resources 486
What You Should Know About Amazon RDS Resource Tags 486
AWS Management Console 487
CLI 491
API 492
Constructing an Amazon RDS Amazon Resource Name (ARN) 493
Related Topics 495
Backing Up and Restoring 495
Working With Automated Backups 496
Creating a DB Snapshot 499
Restoring From a DB Snapshot 501
Copying a DB Snapshot 504
Restoring a DB Instance to a Specified Time 508
Working with Option Groups 510
Option Groups Overview 510
Creating an Option Group 514
Making a Copy of an Option Group 515
Adding an Option to an Option Group 515
Listing the Options and Option Settings for an Option Group 518
Modifying an Option Setting 519
Removing an Option from an Option Group 521
Working with DB Parameter Groups 523
Creating a DB Parameter Group 524
Modifying Parameters in a DB Parameter Group 525
Copying a DB Parameter Group 528
Listing DB Parameter Groups 529
Viewing Parameter Values for a DB Parameter Group 531
DB Parameter Values 534
Working with DB Security Groups 537
Creating a DB Security Group 537
Listing Available DB Security Groups 540
Viewing a DB security group 541
Authorizing Network Access to a DB Security Group from an IP Range 543
Authorizing Network Access to a DB Instance from an Amazon EC2 Instance 545
Revoking Network Access to a DB Instance from an IP Range 547
Related Topics 549
Working with Reserved DB Instances 550
Getting Information About Available Reserved DB Instance Offerings 551
Purchasing a Reserved DB Instance 556
Getting Information About Your Account's Reserved DB Instances 558
Related Topics 561
Using Amazon RDS with Amazon VPC 562
Determining Whether You are Using the EC2-VPC or EC2-Classic Platform 563
Working with a DB Instance in a VPC 564
Working with DB Subnet Groups 565

Hiding a DB Instance in a VPC from the Internet 565
Creating a DB Instance in a VPC 566
Moving a DB Instance not in a VPC into a VPC 568
Using ClassicLink to Connect an EC2-Classic Instance to a DB Instance in an Amazon VPC 569
Monitoring 571
Viewing DB Instance Metrics 572
AWS Management Console 572
CLI 573
API 573
Related Topics 574
Using Amazon RDS Event Notification 575
Amazon RDS Event Categories and Event Messages 576
Subscribing to Amazon RDS Event Notification 580
Listing Your Amazon RDS Event Notification Subscriptions 584
Modifying an Amazon RDS Event Notification Subscription 585
Adding a Source Identifier to an Amazon RDS Event Notification Subscription 587
Removing a Source identifier from an Amazon RDS Event Notification Subscription 588
Listing the Amazon RDS Event Notification Categories 589
Deleting an Amazon RDS Event Notification Subscription 591
Viewing Amazon RDS Events 592
AWS Management Console 592
CLI 592
API 593
Related Topics 593
Database Log Files 594
MySQL Database Log Files 594
Oracle Database Log Files 598
SQL Server Database Log Files 602
PostgreSQL Database Log Files 603
Viewing and Listing Database Log Files 605
Downloading a Database Log File 609
Watching a Database Log File 611
Logging Amazon RDS API Calls Using AWS CloudTrail 615
Configuring CloudTrail Event Logging 615
Amazon RDS Event Entries in CloudTrail Log Files 615
Troubleshooting 618
Cannot Connect to DB Instance 618
Testing the DB Instance Connection 619
Troubleshooting Connection Authentication 619
Security Issues 619
DB Instance Outage or Reboot 620
Parameter Changes Not Taking Effect 620
DB Instance Out of Storage 621
MySQL Issues 622
MySQL Version 5.5.40 Asynchronous I/O Is Disabled 622
Index Merge Optimization Returns Wrong Results 622
Replication Fails After Upgrading to MySQL Version 5.6.21 623
Diagnosing and Resolving Lag Between Read Replicas 624
Diagnosing and Resolving a MySQL Read Replication Failure 625
Creating Triggers with Binary Logging Enabled Requires SUPER Privilege 626
Diagnosing and Resolving Point in Time Restore Failures 627
Oracle GoldenGate Issues 628
Using Oracle GoldenGate with Amazon EC2 Instances 628
Retaining Logs for Sufficient Time 628
Cannot Connect to SQL Server DB Instance 628
Cannot Connect to PostgreSQL DB Instance 629
Amazon RDS API 630

Using the Query API 630
Query Parameters 630
Query Request Authentication 630
Using the SOAP API 633
WSDL and Schema Definitions 633
Programming Language Support 633
Request Authentication 634
Response Structure 635
Web Services References 636
Available Libraries 636
Troubleshooting Applications 636
Retrieving Errors 636
Troubleshooting Tips 637
RDS REST API Reference 637
Related Topics 637
DownloadCompleteDBLogFile 637
Resources 640
Document History 641

What Is Amazon Relational Database Service (Amazon RDS)?

Topics

- Amazon RDS Components (p. 2)
- Available RDS Interfaces (p. 4)
- How You Are Charged for Amazon RDS (p. 5)
- Monitoring an Amazon RDS DB Instance (p. 5)
- What's Next? (p. 5)

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizeable capacity for an industry-standard relational database and manages common database administration tasks.

Why would you want a managed relational database service? Because Amazon RDS takes over many of the difficult or tedious management tasks of a relational database.

- When you buy a server, you get CPU, memory, storage, and IOPS, all bundled together. With Amazon RDS, these are split apart so that you can scale them independently. So, for example, if you need more CPU, less IOPS, or more storage, you can easily allocate them.
- Amazon RDS manages backups, software patching, automatic failure detection, and recovery.
- In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.
- You can have automated backups performed when you need them, or create your own backup snapshot. These backups can be used to restore a database, and the Amazon RDS restore process works reliably and efficiently.
- You can get high availability with a primary instance and a synchronous secondary instance that you can failover to when problems occur. You can also use MySQL or PostgreSQL Read Replicas to increase read scaling.
- You can use the database products you are already familiar with: MySQL, PostgreSQL, Oracle, Microsoft SQL Server, and the new, MySQL-compatible Amazon Aurora DB engine (for information, see Aurora on Amazon RDS (p. 392)).

- In addition to the security in your database package, you can help control who can access your RDS databases by using AWS IAM to define users and permissions. You can also help protect your databases by putting them in a virtual private cloud.

To begin learning more:

- If you are new to RDS but you are familiar with other Amazon Web Services, start with an introduction to the Amazon RDS Components (p. 2). This section discusses the key components of Amazon RDS and how they map to those that you currently work with on your local network.
- For an overview of all AWS products, see What is Cloud Computing?
- Amazon Web Services provides a number of database services. For guidance on which service is best for your environment, see Running Databases on AWS

Amazon RDS Components

Topics

- DB Instances (p. 2)
- Regions and Availability Zones (p. 3)
- Security Groups (p. 3)
- DB Parameter Groups (p. 3)
- DB Option Groups (p. 3)

DB Instances

The basic building block of Amazon RDS is the *DB instance*. A DB instance is an isolated database environment in the cloud. A DB instance can contain multiple user-created databases, and you can access it by using the same tools and applications that you use with a stand-alone database instance. You can create and modify a DB instance by using the Amazon RDS command line interface, the Amazon RDS API, or the AWS Management Console.

Each DB instance runs a *DB engine*. Amazon RDS currently supports the MySQL, PostgreSQL, Oracle, and Microsoft SQL Server DB engines. Each DB engine has its own supported features, and each version of a DB engine may include specific features. Additionally, each DB engine has a set of parameters in a DB parameter group that control the behavior of the databases that it manages.

The computation and memory capacity of a DB instance is determined by its *DB instance class*. You can select the DB instance that best meets your needs. If your needs change over time, you can change DB instances. For information about DB instance classes, see the DB Instance Class section. For pricing information on DB instance classes, go to the Pricing section of the Amazon Relational Database Service (Amazon RDS) product page.

For each DB instance, you can select from 5 GB to 3 TB of associated *storage* capacity. Each DB instance class has minimum and maximum storage requirements for the DB instances that are created from it. It's important to have sufficient storage so that your databases have room to grow and that features for the DB engine have room to write content or log entries.

DB instance storage comes in three types: Magnetic, General Purpose (SSD), and Provisioned IOPS (SSD). They differ in performance characteristics and price, allowing you to tailor your storage performance and cost to the needs of your database. For a complete discussion of the different volume types, see the topic Amazon EBS Volume Types.

You can run a DB instance on a virtual private cloud using Amazon's Virtual Private Cloud (VPC) service. When you use a virtual private cloud, you have control over your virtual networking environment: you can

select your own IP address range, create subnets, and configure routing and access control lists. The basic functionality of Amazon RDS is the same whether it is running in a VPC or not; Amazon RDS manages backups, software patching, automatic failure detection, and recovery. There is no additional cost to run your DB instance in a VPC. For more information on VPC and RDS, see Using Amazon RDS with Amazon Virtual Private Cloud (VPC) (p. 562).

Regions and Availability Zones

Amazon cloud computing resources are housed in highly available data center facilities in different areas of the world (for example, North America, Europe, or Asia). Each data center location is called a region.

Each region contains multiple distinct locations called Availability Zones, or AZs. Each Availability Zone is engineered to be isolated from failures in other Availability Zones, and to provide inexpensive, low-latency network connectivity to other Availability Zones in the same region. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location. For a list of regions and Availability Zones, see Regions and Availability Zones (p. 69).

You can run your DB instance in several Availability Zones, an option called a Multi-AZ deployment. When you select this option, Amazon automatically provisions and maintains a synchronous standby replica of your DB instance in a different Availability Zone. The primary DB instance is synchronously replicated across Availability Zones to the standby replica to provide data redundancy, failover support, eliminate I/O freezes, and minimize latency spikes during system backups.

Security Groups

A security group controls the access to a DB instance. It does so by allowing access to IP address ranges or Amazon EC2 instances that you specify.

Amazon RDS uses DB security groups, VPC security groups, and EC2 security groups. In simple terms, a DB security group controls access to a DB instance that is not in a VPC, a VPC security group controls access to a DB instance inside a VPC, and an Amazon EC2 security group controls access to an EC2 instance and can be used with a DB instance. For more information about security groups, see Amazon RDS Security Groups (p. 110).

DB Parameter Groups

You manage the configuration of a DB engine by using a DB parameter group. A DB parameter group contains engine configuration values that can be applied to one or more DB instances of the same instance type. Amazon RDS applies a default DB parameter group if you don't specify a DB parameter group when you create a DB instance. The default group contains defaults for the specific database engine and instance class of the DB instance.

DB Option Groups

Some DB engines offer tools that simplify managing your databases and making the best use of your data. Amazon RDS makes such tools available through option groups. Currently, option groups are available for Oracle, Microsoft SQL Server, and MySQL 5.6 DB instances. For more information about individual Oracle options, go to Appendix: Options for Oracle Database Engine (p. 226). For more information about SQL Server options, go to Appendix: Options for SQL Server Database Engine (p. 350). For more information about MySQL 5.6 options, go to Appendix: Options for MySQL Database Engine (p. 173). For more information on option groups, go to Working with Option Groups (p. 510).

Available RDS Interfaces

Topics

- Amazon RDS Console (p. 4)
- Command Line Interface (p. 4)
- Programmatic Interfaces (p. 4)

There are several ways that you can interact with Amazon RDS.

Amazon RDS Console

The Amazon RDS console is a simple web-based user interface. From the console, you can perform almost all tasks you need to do from the RDS console with no programming required. To access the Amazon RDS console, sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn//rds/.

Command Line Interface

Amazon RDS provides a Java-based command line interface that gives you access to much of the functionality that is available in the amazon RDS API. For more information, see the Amazon RDS Command Line Toolkit.

Programmatic Interfaces

The following table lists the resources that you can use to access Amazon RDS programmatically.

Resource	Description
AWS SDKs	The AWS SDKs include sample code, libraries, tools, documentation, and templates. To download the AWS SDKs, go to AWS Software Development Kits (SDKs).
Libraries	AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of Amazon Relational Database Service's SOAP and Query APIs. These libraries provide basic functions (not included in Amazon Relational Database Service's SOAP and Query APIs), such as request authentication, request retries, and error handling so you can get started more easily. Libraries and resources are available for the following languages: • Java • PHP • Python • Ruby • Windows and .NET For libraries and sample code in all languages, see Sample Code & Libraries.
Amazon RDS API	If you prefer, you can code directly to the Amazon RDS API. For more information, see Amazon RDS API (p. 630), and see the Amazon Relational Database Service API Reference.

How You Are Charged for Amazon RDS

When you use Amazon RDS, you pay only for what you use, and there are no minimum or setup fees. You are billed according to the following criteria.

- Instance class – Pricing is based on the class (e.g., micro, small, large, xlarge) of the DB instance consumed.
- Running time – You are billed by the instance-hour, which is equivalent to a single instance running for an hour. For example, both a single instance running for two hours and two instances running for one hour consume 2 instance-hours. If a DB instance runs for only part of an hour, you are charged for a full instance-hour.
- Storage – The storage capacity that you have provisioned to your DB instance is billed per GB per month. If you scale your provisioned storage capacity within the month, your bill will be pro-rated.
- I/O requests per month – Total number of storage I/O requests that you have made in a billing cycle.
- Backup storage – Backup storage is the storage that is associated with automated database backups and any active database snapshots that you have taken. Increasing your backup retention period or taking additional database snapshots increases the backup storage consumed by your database. Amazon RDS provides backup storage up to 100% of your provisioned database storage at no additional charge. For example, if you have 10 GB-months of provisioned database storage, we will provide up to 10 GB-months of backup storage at no additional charge. Most databases require less raw storage for a backup than for the primary dataset, so if you don't keep multiple backups, you will never pay for backup storage. Backup storage is free only for active DB instances.
- Data transfer –Internet data transfer in and out of your DB instance.

In addition to regular RDS pricing, you can purchase reserved DB instances. Reserved DB instances let you make a one-time up-front payment for a DB instance and reserve the DB instance for a one- or three-year term at significantly lower rates. For more information on reserved DB instances, see Working with Reserved DB Instances (p. 550)

For Amazon RDS pricing information, see the Amazon RDS product page.

Monitoring an Amazon RDS DB Instance

There are several ways that you can track the performance and health of a DB instance. You can use the free Amazon CloudWatch service to monitor the performance and health of a DB instance; performance charts are shown in the Amazon RDS console. You can subscribe to Amazon RDS events to be notified when changes occur with a DB instance, DB Snapshot, DB parameter group, or DB security group. For more information about Amazon CloudWatch, see Viewing DB Instance Metrics (p. 572). For more information on Amazon RDS event notification, see Using Amazon RDS Event Notification (p. 575)

What's Next?

This section introduced you to the basic infrastructure components that RDS offers. What should you do next?

Getting Started

Create a DB instance using instructions in the Getting Started with Amazon RDS (p. 12) section.

Database Engine Specific Topics

You can review information specific to a particular DB engine in the following sections:

- Oracle on Amazon RDS (p. 194)
- MySQL on Amazon RDS (p. 119)
- Microsoft SQL Server on Amazon RDS (p. 301)
- PostgreSQL on Amazon RDS (p. 355)
- Aurora on Amazon RDS (p. 392)

Setting Up for Amazon RDS

Before you use Amazon RDS for the first time, complete the following tasks:

1. Sign Up for AWS (p. 7)
2. Create an IAM User (p. 8)
3. Determine Requirements (p. 9)
4. Create a Security Group (p. 10)

Sign Up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including Amazon RDS. You are charged only for the services that you use.

With Amazon RDS, you pay only for the resources you use. The Amazon RDS DB instance that you create will be live (not running in a sandbox). You will incur the standard Amazon RDS usage fees for the instance until you terminate it. For more information about Amazon RDS usage rates, see the Amazon RDS product page. If you are a new AWS customer, you can get started with Amazon RDS for free; for more information, see AWS Free Usage Tier.

If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

To create an AWS account

1. Open http://www.amazonaws.cn/, and then click **Sign Up**.
2. Follow the on-screen instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Note your AWS account number, because you'll need it for the next task.

Create an IAM User

Services in AWS, such as Amazon RDS, require that you provide credentials when you access them, so that the service can determine whether you have permission to access its resources. The console requires your password. You can create access keys for your AWS account to access the command line interface or API. However, we don't recommend that you access AWS using the credentials for your AWS account; we recommend that you use AWS Identity and Access Management (IAM) instead. Create an IAM user, and then add the user to an IAM group with administrative permissions or and grant this user administrative permissions. You can then access AWS using a special URL and the credentials for the IAM user.

If you signed up for AWS but have not created an IAM user for yourself, you can create one using the IAM console.

To create the Administrators group

1. Sign in to the AWS Management Console and open the IAM console at https://console.amazonaws.cn/iam/.
2. In the navigation pane, click **Groups**, and then click **Create New Group**.
3. In the **Group Name** box, type `Administrators` , and then click **Next Step**.
4. In the list of policies, select the check box next to the **AdministratorAccess** policy. You can use the **Filter** menu and the **Search** box to filter the list of policies.
5. Click **Next Step**, and then click **Create Group**.

Your new group is listed under **Group Name**.

To create an IAM user for yourself, add the user to the Administrators group, and create a password for the user

1. In the navigation pane, click **Users**, and then click **Create New Users**.
2. In box **1**, type a user name. Clear the check box next to **Generate an access key for each user**. Then click **Create**.
3. In the list of users, click the name (not the check box) of the user you just created. You can use the **Search** box to search for the user name.
4. In the **Groups** section, click **Add User to Groups**.
5. Select the check box next to the **Administrators** group. Then click **Add to Groups**.
6. Scroll down to the **Security Credentials** section. Under **Sign-In Credentials**, click **Manage Password**.
7. Select **Assign a custom password**. Then type a password in the **Password** and **Confirm Password** boxes. When you are finished, click **Apply** .

To sign in as this new IAM user, sign out of the AWS console, then use the following URL, where *your_aws_account_id* is your AWS account number without the hyphens (for example, if your AWS account number is `1234-5678-9012`, your AWS account ID is `123456789012`):

```
https://your_aws_account_id.signin.www.amazonaws.cn/console/
```

Enter the IAM user name and password that you just created. When you're signed in, the navigation bar displays "*your_user_name* @ *your_aws_account_id*".

If you don't want the URL for your sign-in page to contain your AWS account ID, you can create an account alias. From the IAM dashboard, click **Customize** and enter an alias, such as your company name. To sign in after you create an account alias, use the following URL:

```
https://your_account_alias.signin.www.amazonaws.cn/console/
```

To verify the sign-in link for IAM users for your account, open the IAM console and check under **AWS Account Alias** on the dashboard.

Determine Requirements

The basic building block of Amazon RDS is the DB instance. The DB instance is where you create your databases. A DB instance provides a network address called the **Endpoint**. Your applications connect to the endpoint exposed by the DB instance whenever they need to access the databases created in that DB instance. The information you specify when you create the DB instance controls configuration elements such as storage, memory, database engine and version, network configuration, security, and maintenance periods.

You must know your DB instance and network needs before you create a security group and before you create a DB instance. For example, you must know the following:

- What are the memory and processor requirements for your application or service? You will use these settings when you determine what DB instance class you will use when you create your DB instance. For specifications about DB instance classes, see DB Instance Class (p. 65).
- You need to know if your DB instance is going to be in a virtual private cloud (VPC) and then determine what security group rules you will need (you will use these in the next step). The security group rules you need to connect to a DB instance depend on whether your DB instance is in a default VPC, in a user-defined VPC, or outside of a VPC. If you are new user or began using Amazon RDS in the last year, chances are good your account uses a default VPC. For information on determining if your account has a default VPC in a region, see Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 563). The follow list describes the rules for each VPC option:
  - **Default VPC** — If your AWS account has a default VPC in the region, that VPC is configured to support DB instances. If you specify the default VPC when you create the DB instance:
    - You must create a VPC security group that authorizes connections from the application or service to the Amazon RDS DB instance with the database. For information, see Step 4: Creating a VPC Security Group (p. 568).
    - You must specify the default DB subnet group. If this is the first DB instance you have created in the region, Amazon RDS will create the default DB subnet group when it creates the DB instance.
  - **User-defined VPC** — If you want to specify a user-defined VPC when you create a DB instance:
    - You must create a VPC security group that authorizes connections from the application or service to the Amazon RDS DB instance with the database. For information, see Step 4: Creating a VPC Security Group (p. 568)..
    - The VPC must meet certain requirements in order to host DB instances, such as having at least two subnets, each in a separate availability zone. For information, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).
    - You must specify a DB subnet group that defines which subnets in that VPC can be used by the DB instance. For information, see the DB Subnet Group section in Working with a DB Instance in a VPC (p. 564).
  - **No VPC** — if your AWS account does not have a default VPC, and you do not specify a user-defined VPC:
    - You must specify a DB security group that authorizes connections from the devices and Amazon RDS instances running the applications or utilities that will access the databases in the DB instance. For more information, see Working with DB Security Groups (p. 537).
- Do you need failover support? On Amazon RDS, a standby replica of your DB instance that can be used in the event of a failover is called a Multi-AZ deployment. If you have production workloads, you

should use a Multi-AZ deployment. For test purposes, you can usually get by with a single instance, non-Multi-AZ deployment.

- Does your AWS account have policies that grant the permissions needed to perform Amazon RDS operations? If you are connecting to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS operations. For more information, see Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources (p. 88).
- What TCP/IP port will your database be listening on? The firewall at some companies may block connections to the default port for your database engine. If your company firewall blocks the default port, choose another port for the new DB instance. Note that once you create a DB instance that listens on a port you specify, you cannot change the port for the DB instance.
- What region do you want your database in? Having the database close in proximity to the application or web service could reduce network latency.
- What are your storage requirements? Do you need to use Provisioned IOPS? Amazon RDS provides two storage types: Standard and Provisioned IOPS (input/output operations per second). Standard storage offer cost effective storage that is ideal for applications with light or bursty I/O requirements. Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that are sensitive to storage performance and consistency in random access I/O throughput. For more information on Amazon RDS storage, see Storage for Amazon RDS (p. 77).

Once you have the information you need to create the security group and the DB instance, continue to the next step.

Create a Security Group

Security groups act as a firewall for associated DB instances, controlling both inbound and outbound traffic at the instance level. DB instances are created by default with a firewall that prevents access to it. You must therefore add rules to a security group that enable you to connect to your DB instance. Use the network and configuration information you determined in the previous step to create rules to allow access to your DB instance.

The security group you need to create will be either a VPC security group or a DB security group, depending on if the DB instance is going to be in a VPC. If you created your AWS account after March 2013, chances are very good that you have a default VPC, and your DB instance will be created in that VPC. DB instances in a VPC require that you add rules to a VPC security group to allow access to the instance.

For example, if you have an application that will access a database on your DB instance in a VPC, you must add a Custom TCP rule that specifies the port range and IP addresses that application will use to access the database.

To create a VPC security group

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.amazonaws.cn//vpc/.
2. In the top right corner of the AWS Management Console, select the region in which you want to create the VPC security group and the DB instance. In the list of Amazon VPC resources for that region, it should show that you have at least one VPC and several Subnets. If it does not, you do not have a default VPC in that region.
3. In the navigation pane, click **Security Groups**.
4. Click **Create Security Group**.
5. In the **Create Security Group** window, type the name and description of your security group. Select the VPC that you want to create your DB instance in. Click **Yes, Create**.

6. The VPC security group you created should still be selected. The details pane at the bottom of the console window displays the details for the security group, and tabs for working with inbound and outbound rules. Click the **Inbound Rules** tab.
7. On the **Inbound Rules** tab, click **Edit**. Select **Custom TCP Rule** from the **Type** list. Type your port range in the **PortRange** text box, and then type a CIDR value (IP address) in the **Source** text box.
8. If you need to add more IP addresses or different port ranges, click **Add another rule**.
9. If you need to, you can use the **Outbound Rules** tab to add rules for outbound traffic.
10. When you have finished, click **Save**.

    You will use the VPC security group you just created as the security group for your DB instance when you create it. If your DB instance is not going to be in a VPC, then see the topic Working with DB Security Groups (p. 537) to create a DB security group that you will use when you create your DB instance.

    Finally, a quick note about VPC subnets: If you use a default VPC, a default subnet group spanning all of the VPC's subnets has already been created for you. When you use the **Launch a DB Instance** wizard to create a DB instance, you can select the default VPC and use **default** for the **DB Subnet Group**.

    Once you have completed the setup requirements, you can use your requirements and the security group you created to launch a DB instance. For information on creating a DB instance, see the DB engine-specific link in the following list in the Getting Started section of the Amazon RDS User Guide:

    - Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance (p. 12)
    - Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance (p. 18)
    - Creating a SQL Server DB Instance and Connecting to a Database on a SQL Server DB Instance (p. 26)
    - Creating a PostgreSQL DB Instance and Connecting to a Database on a PostgreSQL DB Instance (p. 36)

Getting Started with Amazon RDS

This section shows you how to create and connect to a DB instance using Amazon RDS. You can create, or launch, a DB instance that uses MySQL, Oracle, PostgreSQL, and Microsoft SQL Server.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 7) section before you can create or connect to a DB instance.

Creating a DB instance and connecting to a database on a DB instance is slightly different for each of the DB engines; select the DB engine below that you want to use for detailed information on creating and connecting to the DB instance.

- Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance (p. 12)
- Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance (p. 18)
- Creating a SQL Server DB Instance and Connecting to a Database on a SQL Server DB Instance (p. 26)
- Creating a PostgreSQL DB Instance and Connecting to a Database on a PostgreSQL DB Instance (p. 36)
- Creating a DB Cluster and Connecting to a Database on an Amazon Aurora DB Instance (p. 47)

Once you have created and connected to your DB instance, instructions are provided to help you delete the DB instance.

Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance

The easiest way to create a DB instance is to use the Amazon RDS console. Once you have created the DB instance, you can use standard MySQL utilities such as MySQL Workbench to connect to a database on the DB instance.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 7) section before you can create or connect to a DB instance.

Topics

- Creating a MySQL DB Instance (p. 13)
- Connecting to a Database on a DB Instance Running the MySQL Database Engine (p. 17)

- Deleting a DB Instance (p. 18)

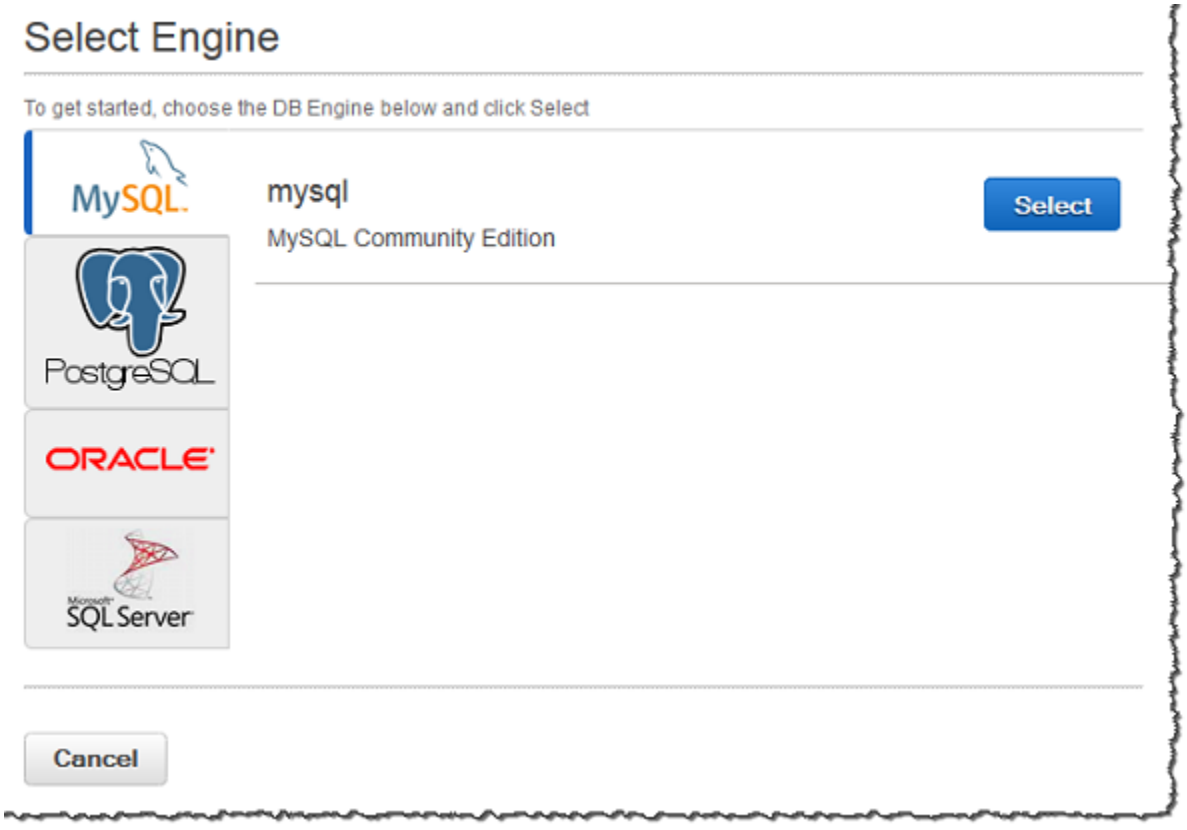
Creating a MySQL DB Instance

The basic building block of Amazon RDS is the DB instance. This is the environment in which you will run your MySQL databases.

In this example, you create a DB instance running the MySQL database engine called *west2-mysql-instance1*, with a *db.m1.small* DB instance class, 5 GB of storage, and automated backups enabled with a retention period of one day.

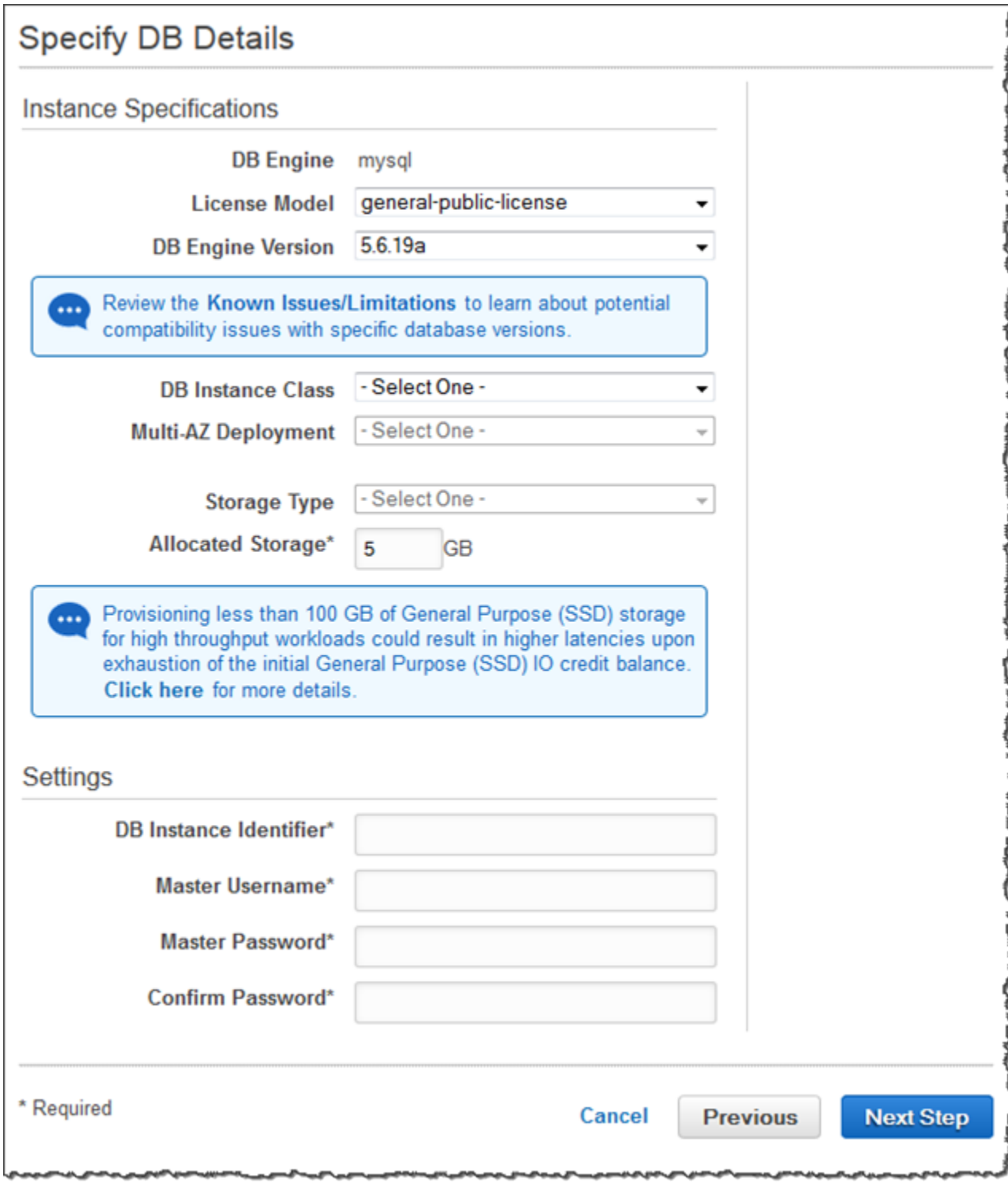
To create a MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the top right corner of the Amazon RDS console, select the region in which you want to create the DB instance.
3. In the navigation pane, click **Instances**.
4. Click **Launch DB Instance**. The **Launch DB Instance Wizard** opens on the **Select Engine** page.



5. On the **Select Engine** page, click the MySQL icon and then click **Select** for the MySQL DB engine.
6. On the **Specify DB Details** page, specify your DB instance information. The following table shows settings for an example DB instance. When the settings are as you want them, click **Next**.

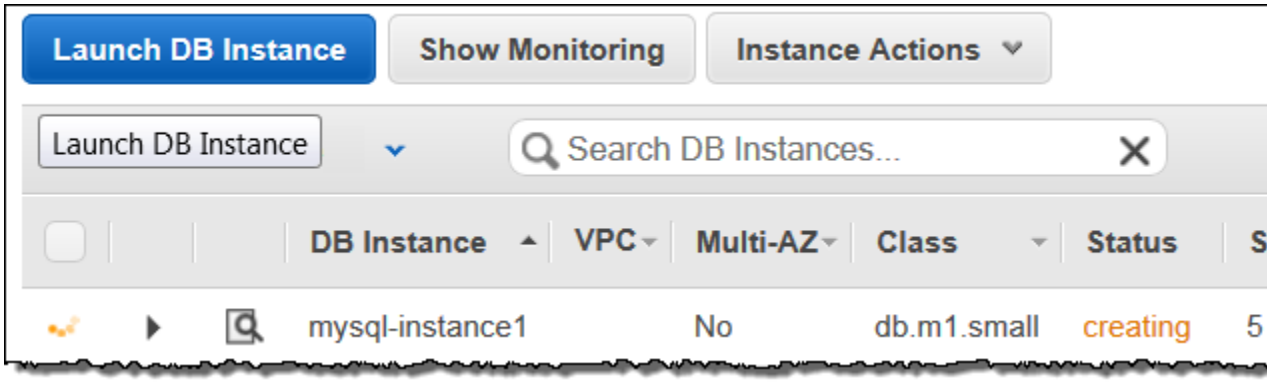
For this parameter...	...Do this:
License Model	Select the default, **`general-public-license`**, to use the general license agreement for MySQL. MySQL has only one license model.
DB Engine Version	Select the default version of MySQL. Note that Amazon RDS supports multiple versions of MySQL in some regions.
DB Instance Class	Select **`db.m1.small`** to select a configuration that equates to 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity.
Multi-AZ Deployment	Select **`No`** to create your DB instance in a single availability zone.
Allocated Storage	Type **`5`** to allocate 5 GB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon Relational Database Service Features.
Storage Type	Select the storage type **`Magnetic`**. For more information about storage, see Storage for Amazon RDS (p. 77).
DB Instance Identifier	Type a name for the DB instance that is unique for your account in the region you selected. You may chose to add some intelligence to the name such as including the region and DB engine you selected, for example **`west2-mysql-instance1`**.
Master Username	Type a name using alphanumeric characters that you will use as the master user name to log on to your DB instance. This will be the user name you use to logon to your database on the DB instance for the first time.
Master Password and **Confirm Password**	Type a password that contains from 8 to 16 printable ASCII characters (excluding /,", and @) for your master user password. This will be the password you will use when you use the user name to logon to your database. Then type the password again in the **Confirm Password** text box.



7. On the **Configure Advanced Settings** page, provide additional information that RDS needs to launch the MySQL DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then click Launch DB Instance.

For this parameter...	...Do this:
VPC	Select the name of the Virtual Private Cloud (VPC) that will host your MySQL DB instance. If your DB instance will not be hosted in a VPC, select **Not in VPC**. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).
Availability Zone	Determine if you want to specify a particular Availability Zone. If you selected **Yes** for the Multi-AZ Deployment parameter on the previous page, you will not have any options here. For more information about Availability Zones, see Regions and Availability Zones (p. 69).
DB Security Groups	Select the security group you want to use with this DB instance. For more information about security groups, see Working with DB Security Groups (p. 537).
Database Name	Type a database name that is 1 to 64 alpha-numeric characters. If you do not provide a name, Amazon RDS will not automatically create a database on the DB instance you are creating.
Database Port	Leave the default value of **3306** unless you have a specific port you want to access the database through. MySQL installations default to port 3306. **Important** You cannot change the port once you create the DB instance, so it is very important that you determine the correct port to use to access the DB instance.
DB Parameter Group	Leave the default value of **default.mysql5.6** unless you created your own DB parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 523).
Option Group	Select the default value of **default:mysql5.6** since this option group is used with the MySQL version you selected on the previous page.
Enable Encryption	Select Yes to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 106).
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For testing purposes, you can set this value to **1**.
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of **No Preference**.
Auto Minor Version Upgrade	Select **Yes** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Maintenance Window	Select the 30 minute window in which pending modifications to your DB instance are applied. If you the time period doesn't matter, select **No Preference**.

8. On the RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to a database on the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new DB instance to become available.



Connecting to a Database on a DB Instance Running the MySQL Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to a database on the DB instance. In this example, you connect to a database on a MySQL DB instance using MySQL monitor commands. One GUI-based application you can use to connect is MySQL Workbench; for more information, go to the Download MySQL Workbench page. For more information on using MySQL, go to the MySQL documentation.

To connect to a database on a DB instance using MySQL monitor

- Type the following command at a command prompt on a client computer to connect to a database on a MySQL DB instance using the MySQL monitor. Substitute the DNS name for your DB instance for <endpoint>, the master user name you used for <mymasteruser>, and the master password you used for <password>.

```
PROMPT> mysql -h <endpoint> -P 3306 -u <mymasteruser> -p
```

You will see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 5.1.32-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Deleting a DB Instance

Once you have connected to the sample DB instance that you created, you should delete the DB instance so you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the **DB Instances** list, select the check box next to the DB instance you wish to delete.
3. Click **Instance Actions**, and then select **Delete** from the dropdown menu.
4. Select **No** in the **Create final Snapshot?** drop-down list box.
5. Click **Yes, Delete**.

Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance

The easiest way to create an Oracle DB instance is to use the RDS console. Once you have created the DB instance, you can use standard Oracle client utilities such as SQL Developer to connect to the instance.

In this example, you create a DB instance running the Oracle database engine called *west2-oracle1*, with a *db.m1.small* DB instance class, 10 GB of storage, and automated backups enabled with a retention period of one day.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 7) section before you can create or connect to a DB instance.

Topics

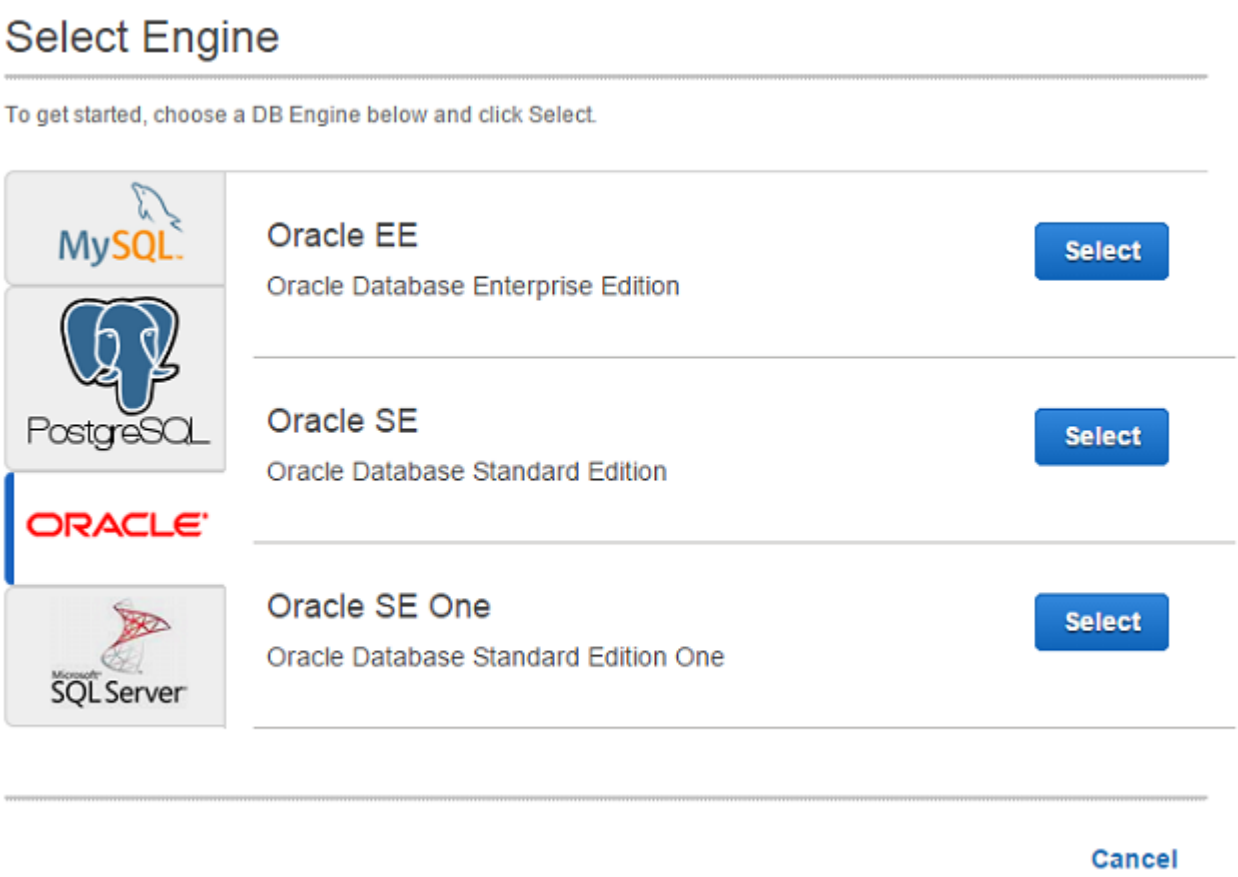
- Creating a DB Instance Running the Oracle Database Engine (p. 18)
- Connecting to a DB Instance Running the Oracle Database Engine (p. 24)
- Deleting a DB Instance (p. 26)

Creating a DB Instance Running the Oracle Database Engine

To launch an Oracle DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the top right corner of the Amazon RDS console, select the region in which you want to create the DB instance.
3. In the navigation pane, click **DB Instances**.
4. Click **Launch DB Instance** to start the **Launch DB Instance Wizard**.

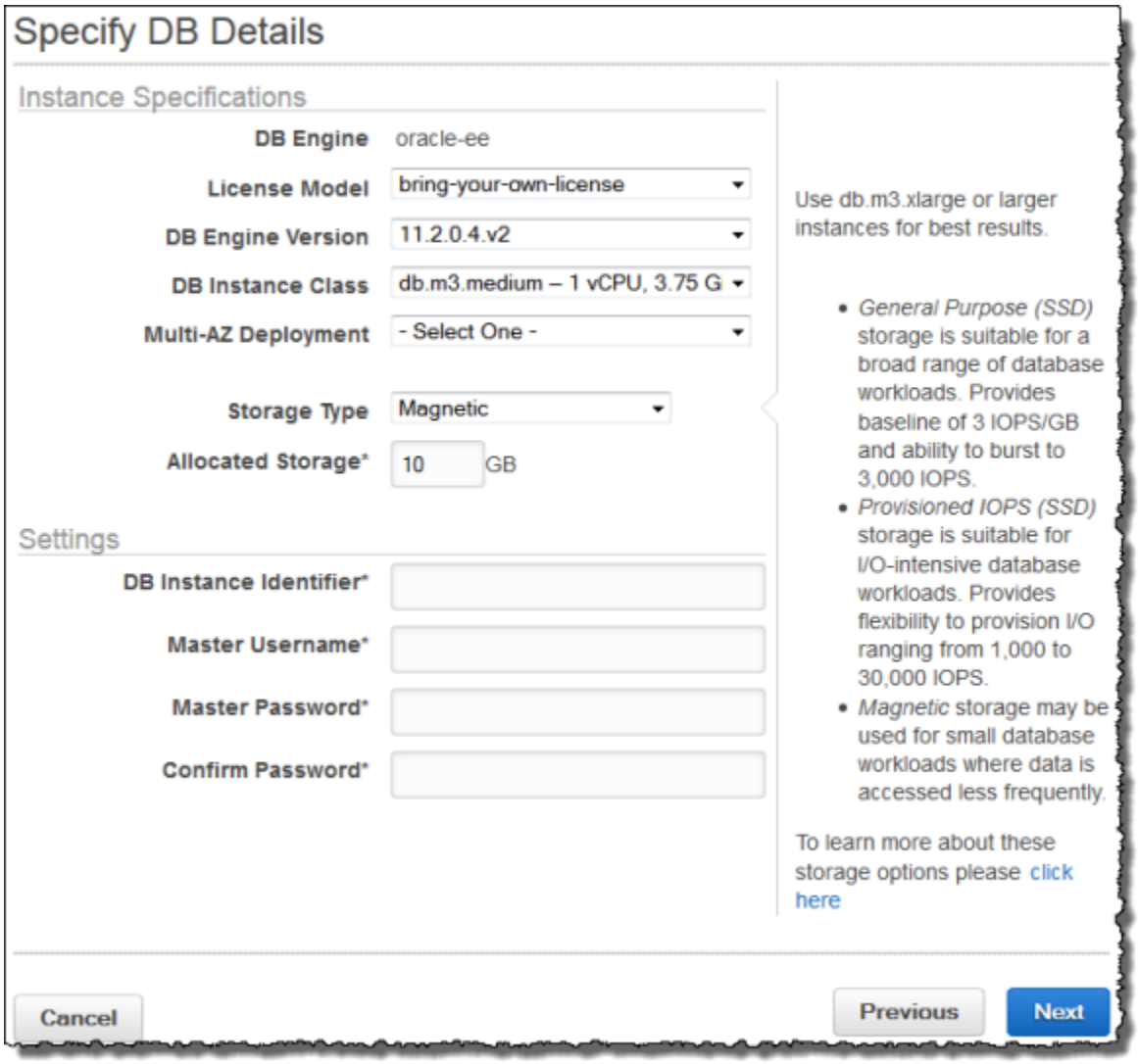
   The wizard opens on the **Select Engine** page.



5. In the **Launch DB Instance Wizard** window, click the Oracle icon, and then click **Select** for the Oracle version you want to use.
6. On the **Production?** page, it asks if you are planning to use the DB instance you are creating for production. If you are, select **Yes**. By selecting **Yes**, the failover option Multi-AZ and the Provisioned IOPS storage option will be preselected in the following step. Click **Next** to continue.
7. On the **Specify DB Details** page, specify your DB instance information. The following table shows settings for an example DB instance. Click **Next** when you are finished.

For this parameter...	...Do this:
License Model	Select **`bring-your-own-license`**, to provide your own license for using Oracle. Some regions support additional licensing options for Oracle.
DB Engine Version	Select the default version of Oracle.
DB Instance Class	Select **`db.m3.medium`** to select a configuration that equates to 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity.
Multi-AZ Deployment	Select **`No`** to create your DB instance in a single availability zone.

For this parameter...	...Do this:
Allocated Storage	Type **`10`** to allocate 10 GB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon Relational Database Service Features.
Storage Type	Select the storage type **`Magnetic`**. For more information about storage, see Storage for Amazon RDS (p. 77).
DB Instance Identifier	Type a name for the DB Instance that is unique for your account in the region you selected. You may choose to add some intelligence to the name such as including the region and DB engine you selected, for example **`oracle-unstance1`**.
Master User Name	Type a name that you will use as the master user name to log on to your DB instance with all database privileges. This user account is used to log into the DB instance and is granted the "DBA" role.
Master User Password and **Confirm Password**	Type a password that contains from 8 to 30 printable ASCII characters (excluding /,", and @) for your master user password, and then type the password again in the Confirm Password text box.


Specify DB Details
Instance Specifications
DB Engine
oracle-ee
License Model
bring-your-own-license
DB Engine Version
11.2.0.4.v2
DB Instance Class
db.m3.medium – 1 vCPU, 3.75 G
Multi-AZ Deployment
- Select One -
Storage Type
Magnetic
Allocated Storage*
10
GB
Settings
DB Instance Identifier*
Master Username*
Master Password*
Confirm Password*
Use db.m3.xlarge or larger instances for best results.
General Purpose (SSD) storage is suitable for a broad range of database workloads. Provides baseline of 3 IOPS/GB and ability to burst to 3,000 IOPS.
Provisioned IOPS (SSD) storage is suitable for I/O-intensive database workloads. Provides flexibility to provision I/O ranging from 1,000 to 30,000 IOPS.
Magnetic storage may be used for small database workloads where data is accessed less frequently.
To learn more about these storage options please click here
Cancel
Previous
Next

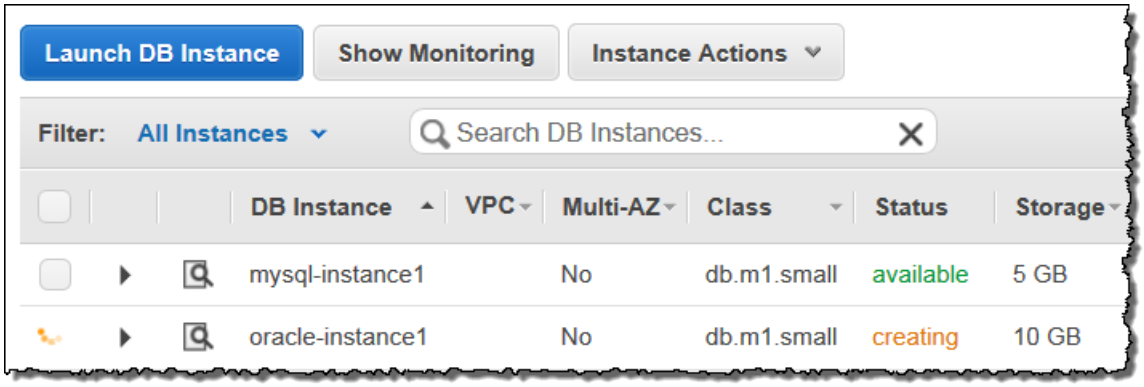
8. On the **Configure Advanced Settings** page, provide additional information that RDS needs to launch the Oracle DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then click Launch DB Instance.

For this parameter...	...Do this:
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, select the default VPC. If you are creating a DB instance on the previous E2-Classic platform, select **`Not in VPC`**. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).

For this parameter...	...Do this:
DB Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, select **default**, which will be the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, select the DB subnet group you created for that VPC. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).
Publicly Accessible	Select **Yes** to give the DB instance a public IP address, meaning that it will be accessible outside the VPC; otherwise, select **No**, so the DB instance will only be accessible from inside the VPC. For more information about hiding DB instances from public access, see Hiding a DB instance in a VPC from the Internet.
Availability Zone	Use the default of **No Preference**.
VPC Security Group	If you are a new customer to AWS, select the default VPC. If you have created your own VPC security group, select the VPC security group you previously created.
Database Name	Type a name for your database that begins with a letter and contains up to 8 alpha-numeric characters. If you do not provide a name, Amazon RDS will not create a database on the DB instance you are creating. The default database name is **ORCL**.
Database Port	Use the default value of **1521** unless you have a specific port you want to access the database through. Oracle installations default to port 1521, but some firewalls block this port by default. If you are unsure, ask your system administrator what port you should use. **Important** You cannot change the port once you create the DB instance, so it is very important that you determine the correct port to use to access the DB instance.
Parameter Group	Use the default value of **default.oracle-ee-11.2**.
Option Group	Select the default value of **default:oracle-ee-11-2**.
Character Set Name	Select the default value of **AL32UTF8** for the Unicode 5.0 UTF-8 Universal character set. Note that you cannot change the character set after the DB instance is created.
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For testing purposes, you can set this value to **1**.
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of **No Preference**.
Auto Minor Version Upgrade	Select **Yes** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.

For this parameter...	...Do this:
Maintenance Window	Select the 30 minute window in which pending modifications to your DB instance are applied. If you the time period doesn't matter, select **`No Preference`**.

9. On the final page of the wizard, click **Close**.
10. On the RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new instance to be available.

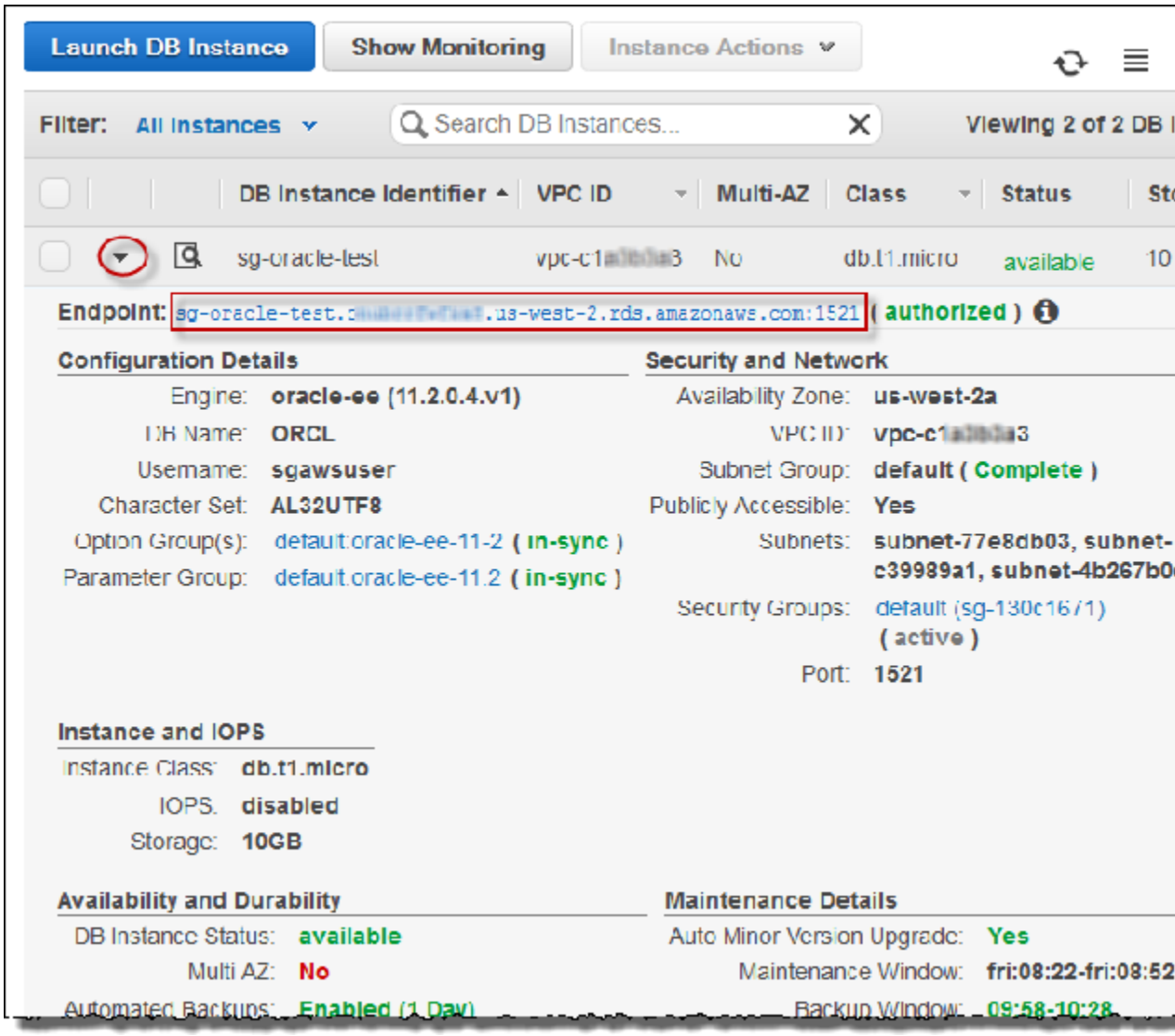


Connecting to a DB Instance Running the Oracle Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. In this example, you connect to a DB instance running the Oracle database engine using the Oracle command line tools. For more information on using Oracle, go to the Oracle website.

This example uses the Oracle *sqlplus* command line utility. This utility is part of the Oracle software distribution. To download a stand-alone version of this utility, go to the SQL*Plus User's Guide and Reference.

1. Open the RDS console, then select **Instances** in the left column to display a list of your DB instances.
2. In the row for your Oracle DB instance, select the arrow to display the summary information for the instance.
3. The **Endpoint** field contains part of the connection information for your DB instance. The **Endpoint** field has two parts separated by a colon (:). The part before the colon is the DNS name for the instance, the part following the colon is the port.



4. Type the following command on one line at a command prompt to connect to a DB instance using the sqlplus utility. The value for `Host` will be the DNS name for your DB instance, the value for `Port` will be the port you assigned the DB instance, and the value for the Oracle `SID` will be the name of the DB instance's database that you specified when you created the DB instance, not the name of the DB instance.

```
PROMPT>sqlplus 'mydbusr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<endpoint>)
    (PORT=<port number>))(CONNECT_DATA=(SID=<database name>)))'
```

You will see output similar to the following.

```
SQL*Plus: Release 11.1.0.7.0 - Production on Wed May 25 15:13:59 2011

SQL>
```

Deleting a DB Instance

Once you have connected to the sample DB instance that you created, you should delete the DB instance so you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the **DB Instances** list, select the check box next to the DB instance you wish to delete.
3. Click **Instance Actions**, and then select **Delete** from the dropdown menu.
4. Select **No** in the **Create final Snapshot?** drop-down list box.
5. Click **Yes, Delete**.

Creating a SQL Server DB Instance and Connecting to a Database on a SQL Server DB Instance

The easiest way to create a DB instance is to use the RDS console. Once you have created the DB instance, you can use standard SQL Server utilities to connect to the DB instance such as the Microsoft SQL Server Management Studio utility.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 7) section before you can create or connect to a DB instance.

Topics

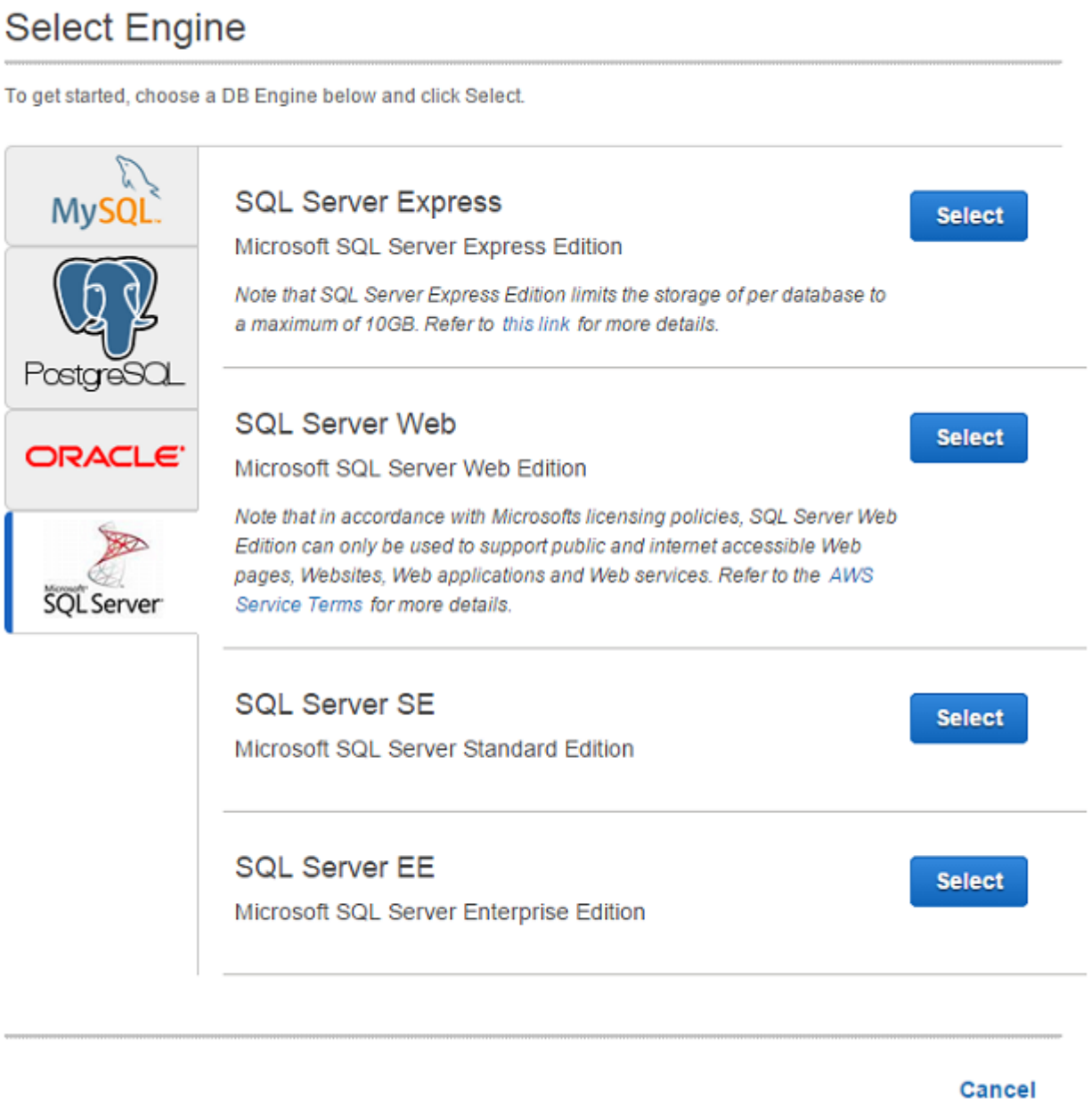
- Creating a SQL Server DB Instance (p. 26)
- Connecting to a SQL Server DB Instance Using SQL Server Management Studio (p. 31)
- Troubleshooting a Connection to a DB Instance Running SQL Server (p. 35)
- Deleting a DB Instance (p. 36)

Creating a SQL Server DB Instance

To create a DB instance running the Microsoft SQL Server DB engine

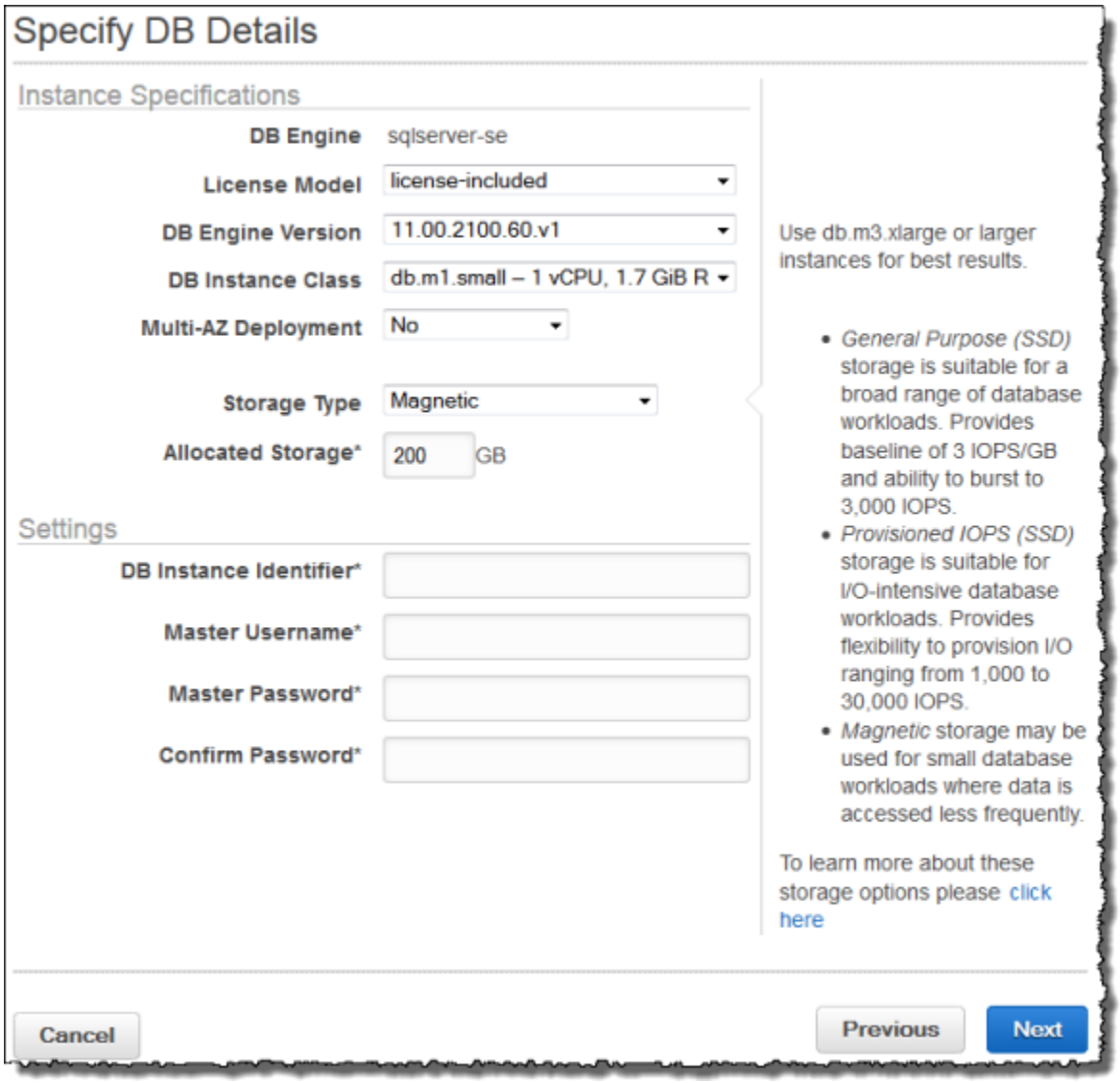
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the top right corner of the Amazon RDS console, select the region in which you want to create the DB instance.
3. In the navigation pane, click **Instances**.
4. Click **Launch DB Instance** to start the **Launch DB Instance Wizard**.

   The wizard opens on the **Select Engine** page.


Select Engine
To get started, choose a DB Engine below and click Select.
MySQL
PostgreSQL
ORACLE
Microsoft SQL Server
SQL Server Express
Microsoft SQL Server Express Edition
Note that SQL Server Express Edition limits the storage of per database to a maximum of 10GB. Refer to this link for more details.
Select
SQL Server Web
Microsoft SQL Server Web Edition
Note that in accordance with Microsofts licensing policies, SQL Server Web Edition can only be used to support public and internet accessible Web pages, Websites, Web applications and Web services. Refer to the AWS Service Terms for more details.
Select
SQL Server SE
Microsoft SQL Server Standard Edition
Select
SQL Server EE
Microsoft SQL Server Enterprise Edition
Select
Cancel

5. In the **Launch DB Instance Wizard** window, click the SQL Server icon, then click **Select** for the SQL Server version you want to use.
6. On the **Production?** page, it asks if you are planning to use the DB instance you are creating for production. If you are, select **Yes**. By selecting **Yes**, the failover option Multi-AZ and the Provisioned IOPS storage option will be preselected in the following step. Click **Next** to continue.
7. On the **Specify DB Details** page, specify your DB instance information. The following table shows settings for an example DB instance using SQL Server Standard Edition. Click **Next** when you are finished.

For this parameter...	...Do this:
License Model	Select, **`license-included`**, to use the general license agreement for Microsoft SQL Server.
DB Engine Version	Select the default version of SQL Server.
DB Instance Class	Select **`db.m1.small`** to select a configuration that equates to 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity. For more information about all the DB instance class options, see DB Instance Class (p. 65).
Multi-AZ Deployment	Select **`No`** to create your DB instance in a single availability zone.
Allocated Storage	Type **`200`** to allocate 200 GB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon Relational Database Service Features.
Storage Type	Select the storage type **`Magnetic`**. For more information about storage, see Storage for Amazon RDS (p. 77).
DB Instance Identifier	Type a name for the DB instance of 15 alphanumeric characters or less that is unique for your account in the region you selected. You may chose to add some intelligence to the name such as including the region and DB Engine you selected, such as **`sqlsv-instance1`**.
Master Username	Type a name that you will use as the master username to log on to your DB Instance with all database privileges. The master username is a SQL Server Authentication login that is a member of the processadmin, public, and setupadmin fixed server roles.
Master Password and **Confirm Password**	Type a password that contains from 8 to 128 printable ASCII characters (excluding /,", and @) for your master user password, and then type it again in the **Confirm Password** text box.

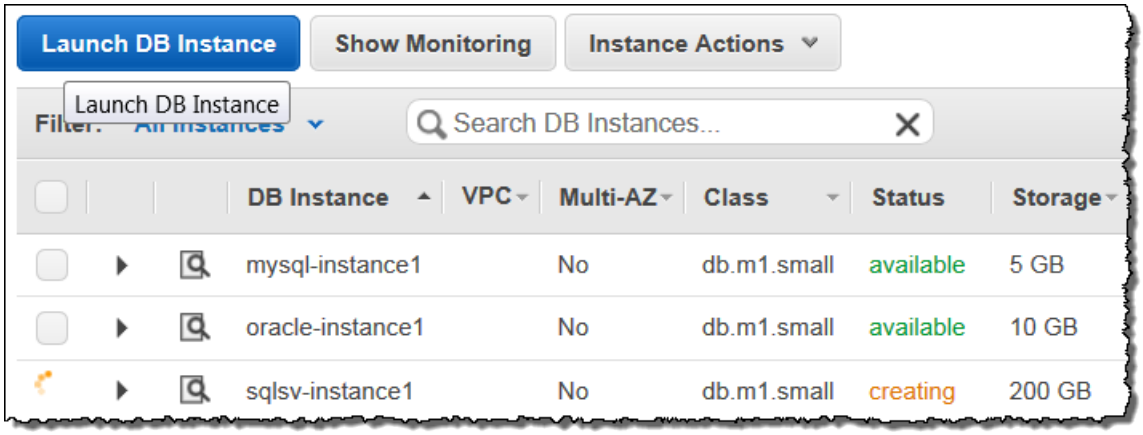

Specify DB Details
Instance Specifications
DB Engine
sqlserver-se
License Model
license-included
DB Engine Version
11.00.2100.60.v1
DB Instance Class
db.m1.small – 1 vCPU, 1.7 GiB R
Multi-AZ Deployment
No
Storage Type
Magnetic
Allocated Storage*
200
GB
Settings
DB Instance Identifier*
Master Username*
Master Password*
Confirm Password*
Use db.m3.xlarge or larger instances for best results.
General Purpose (SSD) storage is suitable for a broad range of database workloads. Provides baseline of 3 IOPS/GB and ability to burst to 3,000 IOPS.
Provisioned IOPS (SSD) storage is suitable for I/O-intensive database workloads. Provides flexibility to provision I/O ranging from 1,000 to 30,000 IOPS.
Magnetic storage may be used for small database workloads where data is accessed less frequently.
To learn more about these storage options please click here
Cancel
Previous
Next

8. On the **Configure Advanced Settings** page, provide additional information that Amazon RDS needs to launch the SQL Server DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then click Launch DB Instance.

For this parameter...	...Do this:
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, select the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, select **`Not in VPC`**. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).

For this parameter...	...Do this:
DB Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, select **`default`**, which will be the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, select the DB subnet group you created for that VPC. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).
Publicly Accessible	Select **`Yes`** to give the DB instance a public IP address, meaning that it will be accessible outside the VPC; otherwise, select **`No`**, so the DB instance will only be accessible from inside the VPC. For more information about hiding DB instances from public access, see Hiding a DB instance in a VPC from the Internet.
Availability Zone	Use the default value of **`No Preference`** unless you want to specify an Availability Zone.
VPC Security Group	If you are a new customer to AWS, select the default VPC. Otherwise, select the VPC security group you previously created.
Database Port	Leave the default value of **`1433`** unless you have a specific port you want to access the database through. SQL Server installations default to port 1433, but in some cases a firewall may block this port. If in doubt, ask your network administrator what port you should use. **Important** You cannot change the port once you create the DB instance, so it is very important that you determine the correct port to use to access the DB instance.
Parameter Group	Use the default value unless you have created your own parameter group.
Option Group	Use the default value unless you have created your own option group.
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For testing purposes, you can set this value to **`1`**.
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of **`No Preference`**.
Auto Minor Version Upgrade	Select **`Yes`** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Maintenance Window	Select the 30 minute window in which pending modifications to your DB instance are applied. If you the time period doesn't matter, select **`No Preference`**.

9. On the final page of the wizard, click **Close**.
10. On the RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new instance to be available.

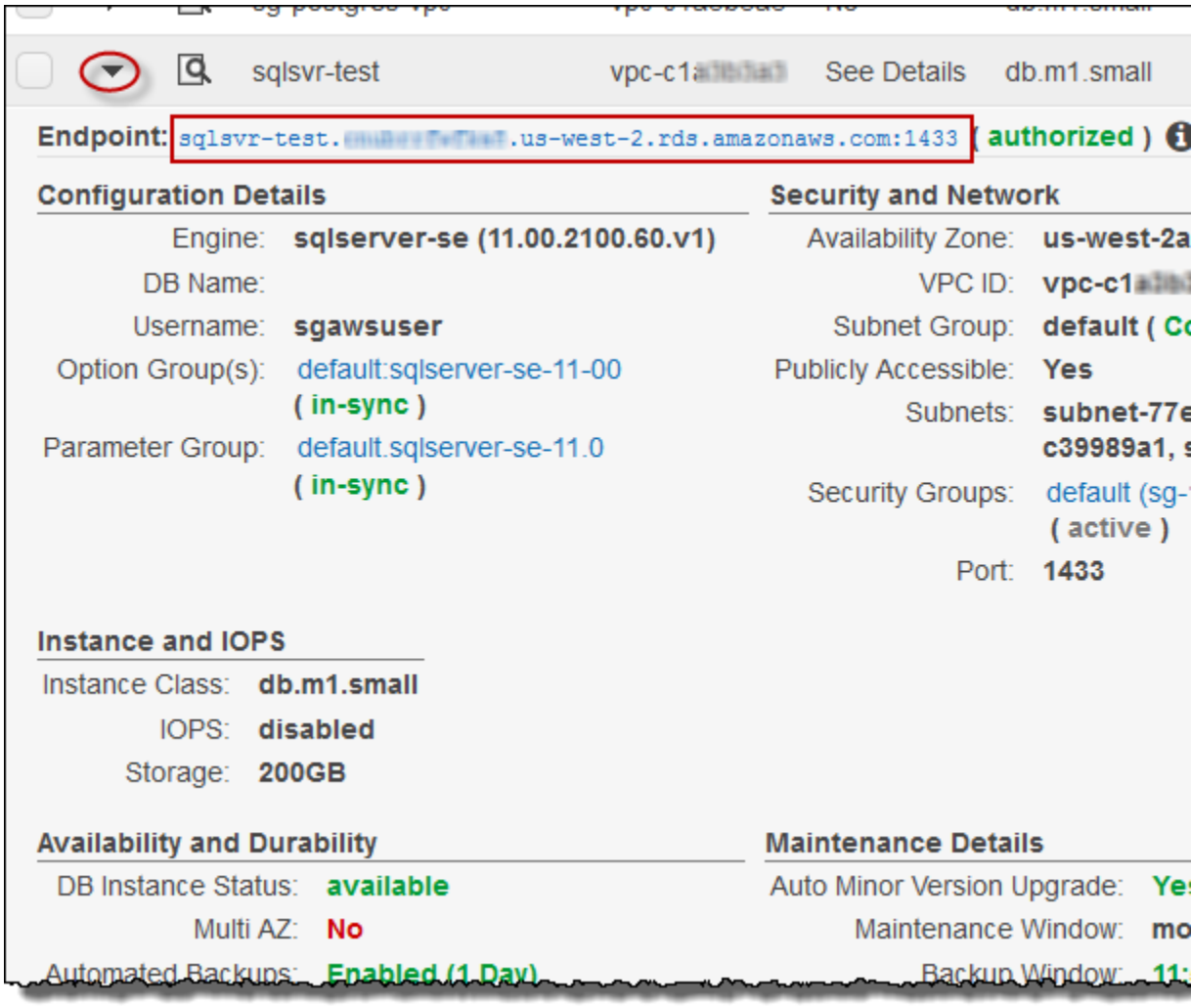


Connecting to a SQL Server DB Instance Using SQL Server Management Studio

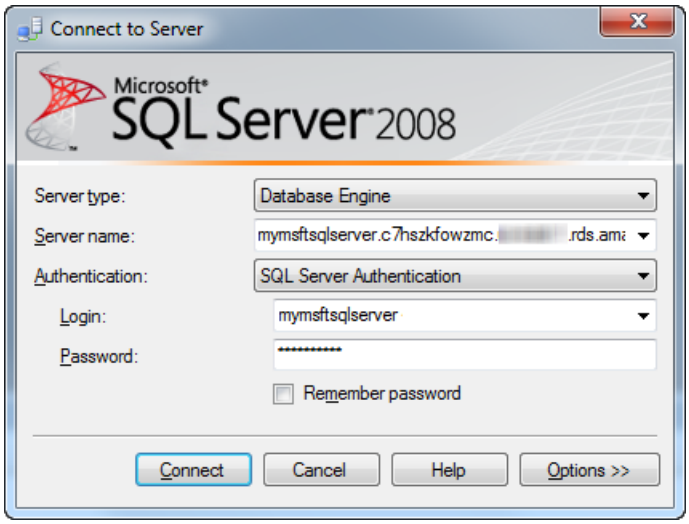
This example uses the Microsoft SQL Server Management Studio utility. This utility is part of the Microsoft SQL Server software distribution. To download a stand-alone version of this utility, go to the Microsoft Download Center - Microsoft SQL Server Management Studio Express.

To connect to a DB Instance using Microsoft SQL Server Management Studio

1. Find the DNS name and port for your DB Instance.
   a. Open the RDS console, then select **Instances** in the left column to display a list of your DB instances.
   b. In the row for your SQL Server DB instance, select the arrow to display the summary information for the instance.
   c. The **Endpoint** field has two parts separated by a colon (:). The part before the colon is the DNS name for the instance, the part following the colon is the port.



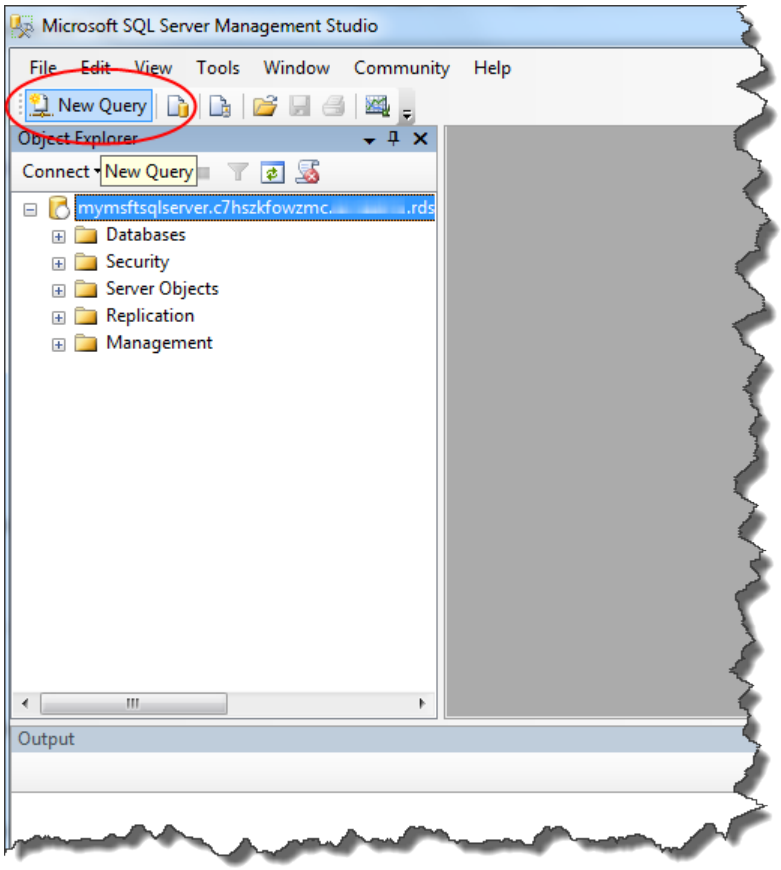
2. Run Microsoft SQL Server Management Studio.
3. The **Connect to Server** dialog box appears.



4. In the **Server type:** drop-down list box, select **`Database Engine`**.
5. In the **Server name:** text field, enter or paste the DNS name of the DB Instance running the Microsoft SQL Server database engine, followed by a comma and then the port number of the DB Instance. For example, the **Server name** could be: **`sqlsv-instance1.cg034hpkmmjt.us-east-1.rds.amazonaws.com,1433`**.
6. From the **Authentication** drop-down list box, select **`SQL Server Authentication`**.
7. Enter the master user name for the DB Instance in the **Login:** text box.
8. Enter the password for the master user in the **Password:** text box.
9. Click the **Connect** button.

   After a few moments, Microsoft SQL Server Management Studio should be connected to your DB Instance.
10. Click the **New Query** button at the top left of the SQL Server Management Studio window.

    A new SQL Query window will open.

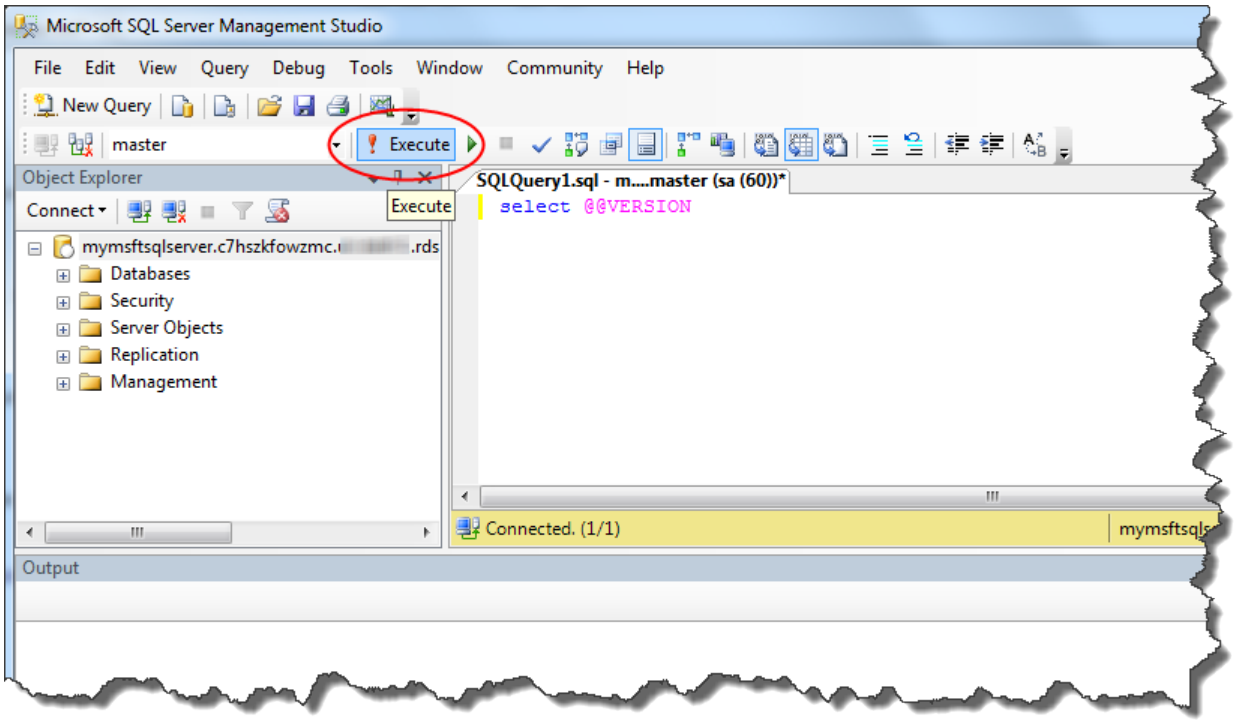


11. Type the following SQL query:

```
select @@VERSION
```

12. Click the **! Execute** button on the SQL Enterprise Manager toolbar to run the query.

    You should see a version string returned from your Microsoft SQL Server DB Instance displayed in the output window.



Troubleshooting a Connection to a DB Instance Running SQL Server

There are several common causes for problems when trying to connect to a DB instance using SQL Server Management Studio:

- The access rules enforced by your local firewall and the IP addresses you authorized to access your DB instance in the instance's security group are not in sync. If you used Microsoft SQL Server Management Studio and you followed the settings specified in the steps above and you are unable to connect, the problem is most likely the egress or ingress rules on your firewall. For more information about security groups, see Amazon RDS Security Groups (p. 110).
- If you cannot send out or receive communications over the port you specified when you created the DB instance, you will not be able to connect to the DB instance. Check with your network administrator to determine if the port you specified for your DB instance is allowed to be used for inbound and outbound communication.
- For newly created DB instances, you must wait for the DB instance status to be "Available" before you can connect to the instance. Depending on the size of your DB instance, it can take up to 20 minutes before the instance is available.

Here are a few things to check if you know that you can send and receive communications through your firewall for the port you specified when you created the DB instance.

- **Could not open a connection to SQL Server - Microsoft SQL Server, Error: 53** - You must include the port number when you specify the Server Name when using Microsoft SQL Server Management Studio. For example, the server name for a DB instance (including the port number) could be: **`sqlsvr-pdz.c6c8mdfntzgv0.region.rds.amazonaws.com,1433`**.

- **No connection could be made because the target machine actively refused it - Microsoft SQL Server, Error: 10061** - You were able to reach the DB instance but the connection was refused. This is often caused by the user name or password being incorrect.

Deleting a DB Instance

Once you have connected to the sample DB instance that you created, you should delete the DB instance so you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the **DB Instances** list, select the check box next to the DB instance you wish to delete.
3. Click **Instance Actions**, and then select **Delete** from the dropdown menu.
4. Select **No** in the **Create final Snapshot?** drop-down list box.
5. Click **Yes, Delete**.

Creating a PostgreSQL DB Instance and Connecting to a Database on a PostgreSQL DB Instance

The easiest way to create a DB instance is to use the RDS console. Once you have created the DB instance, you can use standard SQL client utilities to connect to the DB instance such as the pgAdmin utility. In this example, you create a DB instance running the PostgreSQL database engine called west2-postgres1, with a db.m1.small DB instance class, 10 GB of storage, and automated backups enabled with a retention period of one day.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 7) section before you can create or connect to a DB instance.

Topics

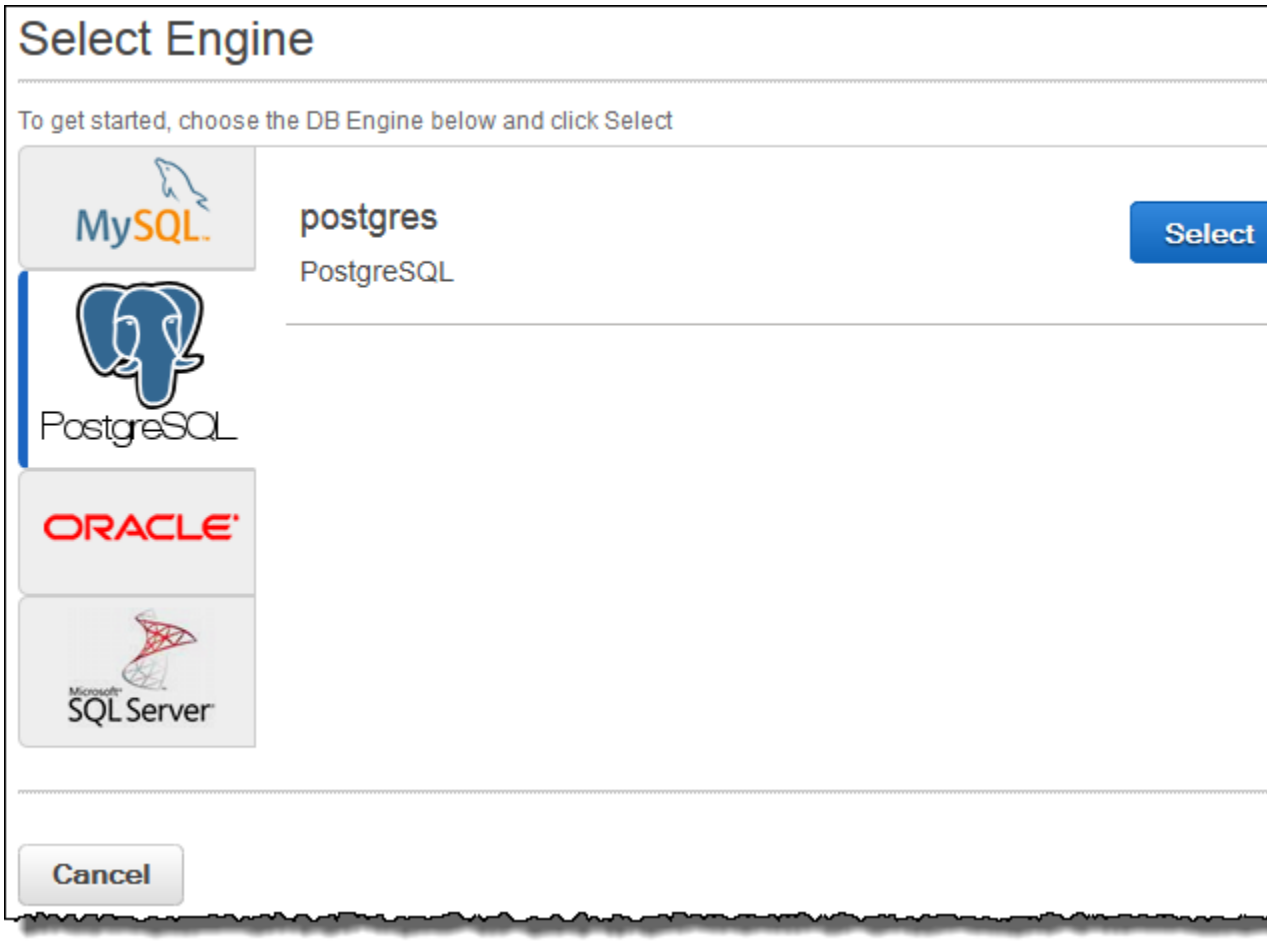
- Creating a PostgreSQL DB Instance (p. 36)
- Connecting to a PostgreSQL DB Instance (p. 43)
- Deleting a DB Instance (p. 46)

Creating a PostgreSQL DB Instance

To create a DB Instance Running the PostgreSQL DB Engine

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the top right corner of the AWS Management Console, select the region in which you want to create the DB instance.
3. In the navigation pane, click **Instances**.
4. Click **Launch DB Instance** to start the **Launch DB Instance Wizard**.

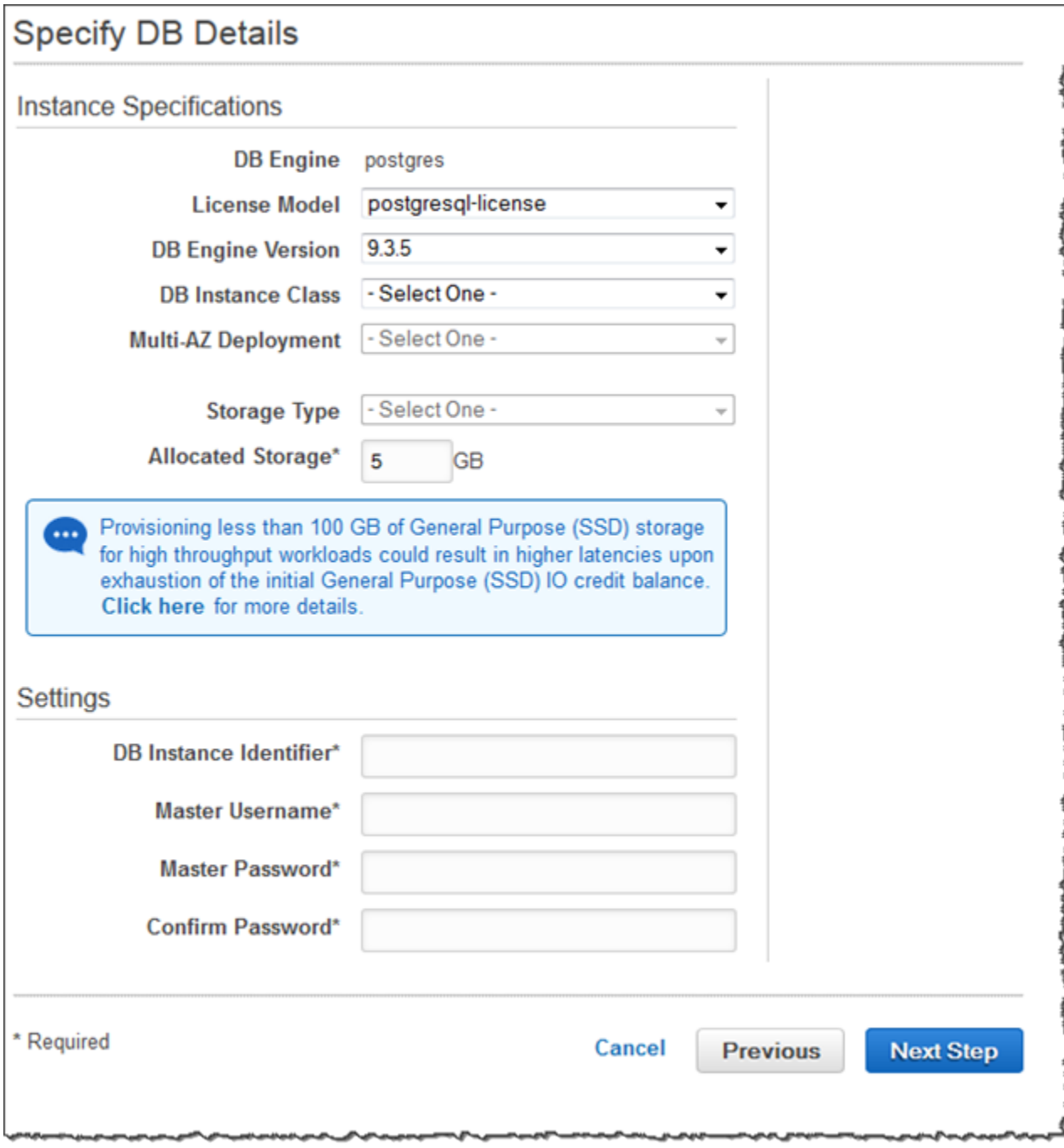
   The wizard opens on the **Select Engine** page.



5. On the **Select Engine** page, click the PostgreSQL icon, and then click **Select**.
6. Next, the **Production?** page asks if you are planning to use the DB instance you are creating for production. If you are, select **Yes**. By selecting **Yes**, the failover option **Multi-AZ** and the **Provisioned IOPS** storage option will be preselected in the following step. Click **Next** when you are finished.
7. On the **Specify DB Details** page, specify your DB instance information. Click **Next** when you are finished.

For this parameter...	...Do this:
License Model	PostgreSQL has only one license model. Select the default, **`postgresql-license`**, to use the general license agreement for PostgreSQL.
DB Engine Version	Select the version of PostgreSQL you want to use.
DB Instance Class	Select **`db.m1.small`** to select a configuration that equates to 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity. For more information about all the DB instance class options, see DB Instance Class (p. 65).
Multi-AZ Deployment	Select **`No`** to create your DB instance in a single availability zone . For more information about multiple Availability Zones, see Regions and Availability Zones (p. 69).

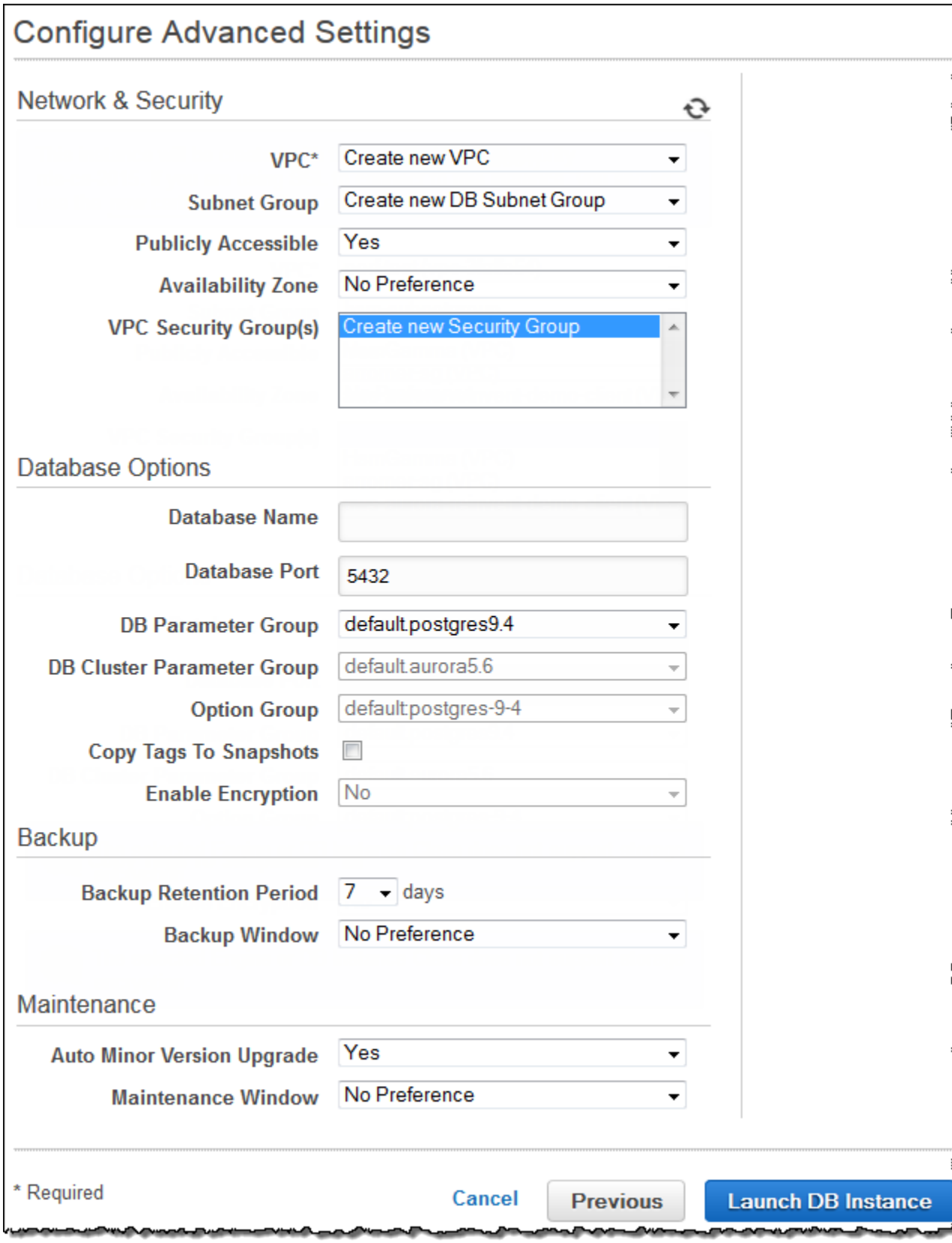
For this parameter...	...Do this:
Allocated Storage	Type **5** to allocate 5 GB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon Relational Database Service Features.
Storage Type	Select the storage type **`Magnetic`**. For more information about storage, see Storage for Amazon RDS (p. 77).
DB Instance Identifier	Type a name for the DB instance that is unique for your account in the region you selected. You may chose to add some intelligence to the name such as including the region and DB engine you selected, for example **`postgreSQL-test`**.
Master Username	Type a name using alphanumeric characters that you will use as the master user name to log on to your DB instance. For information on the default privileges granted to the master user name, see Amazon RDS PostgreSQL Planning Information (p. 356)
Master Password and **Confirm Password**	Type a password that contains from 8 to 128 printable ASCII characters (excluding /,", and @) for your master password, then type the password again in the **Confirm Password** text box.
Enable Encryption	Select to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 106).



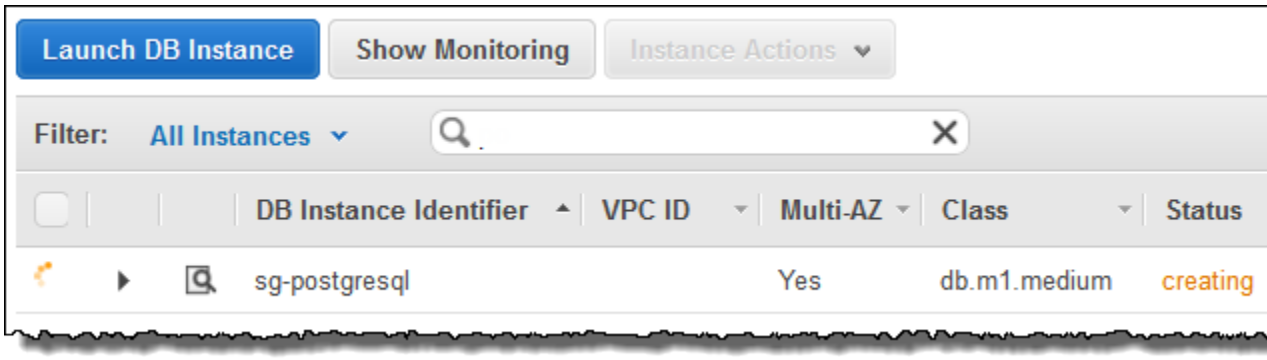
8. On the **Configure Advanced Settings** page, provide additional information that RDS needs to launch the PostgreSQL DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then click Launch DB Instance.

For this parameter...	...Do this:
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, select the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, select **`Not in VPC`**. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).
DB Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, select **`default`**, which will be the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, select the DB subnet group you created for that VPC. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).
Publicly Accessible	Select **`Yes`** to give the DB instance a public IP address, meaning that it will be accessible outside the VPC; otherwise, select **`No`**, so the DB instance will only be accessible from inside the VPC. For more information about hiding DB instances from public access, see Hiding a DB instance in a VPC from the Internet.
Availability Zone	Use the default value of **`No Preference`** unless you want to specify an Availability Zone.
VPC Security Group	If you are a new customer to AWS, select the default VPC. If you created a VPC security group, select the VPC security group you previously created.
Database Name	Type a name for your database of up to 63 alpha-numeric characters. If you do not provide a name, the default "postgres" database is created.
Database Port	Specify a port you want to use to access the database. PostgreSQL installations default to port 5432 . **Important** You cannot change the port once you create the DB instance, so it is very important that you determine the correct port to use to access the DB instance.
Parameter Group	Use the default value unless you have created your own parameter group.
Option Group	Use the default value unless you have created your own option group.
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For testing purposes, you can set this value to **`1`**.
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of **`No Preference`**.

For this parameter...	...Do this:
Auto Minor Version Upgrade	Select **`Yes`** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Maintenance Window	Select the 30 minute window in which pending modifications to your DB instance are applied. If you the time period doesn't matter, select **`No Preference`**.


Configure Advanced Settings
Network & Security
VPC*
Create new VPC
Subnet Group
Create new DB Subnet Group
Publicly Accessible
Yes
Availability Zone
No Preference
VPC Security Group(s)
Create new Security Group
Database Options
Database Name
Database Port
5432
DB Parameter Group
default.postgres9.4
DB Cluster Parameter Group
default.aurora5.6
Option Group
default:postgres-9-4
Copy Tags To Snapshots
Enable Encryption
No
Backup
Backup Retention Period
7
days
Backup Window
No Preference
Maintenance
Auto Minor Version Upgrade
Yes
Maintenance Window
No Preference
* Required
Cancel
Previous
Launch DB Instance

9. On the final page of the wizard, click **Close**.
10. On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new instance to be available.



Connecting to a PostgreSQL DB Instance

After Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. It is important to note that the security group you assigned to the DB instance when you created it must allow access to the DB instance. If you have difficulty connecting to the DB instance, the problem is most often with the access rules you set up in the security group you assigned to the DB instance.

This section shows two ways to connect to a PostgreSQL DB instance. The first example uses *pgAdmin*, a popular Open Source administration and development tool for PostgreSQL. You can download and use *pgAdmin* without having a local instance of PostgreSQL on your client computer. The second example uses *psql*, a command line utility that is part of a PostgreSQL installation. To use *psql*, you must have a PostgreSQL installed on your client computer or have installed the *psql* client on your machine.

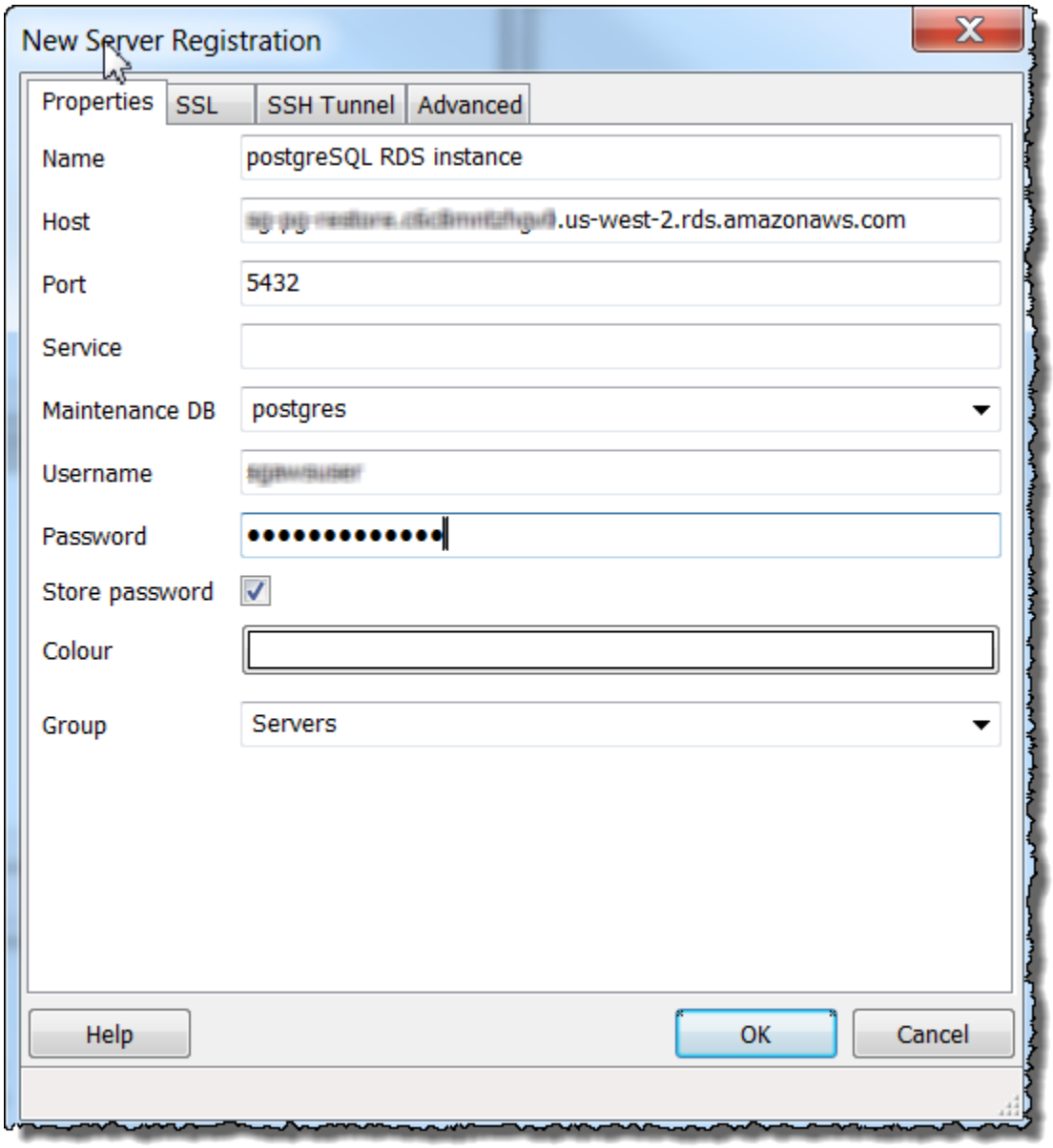
In this example, you connect to a PostgreSQL DB instance using pgAdmin.

Using pgAdmin to Connect to a PostgreSQL DB Instance

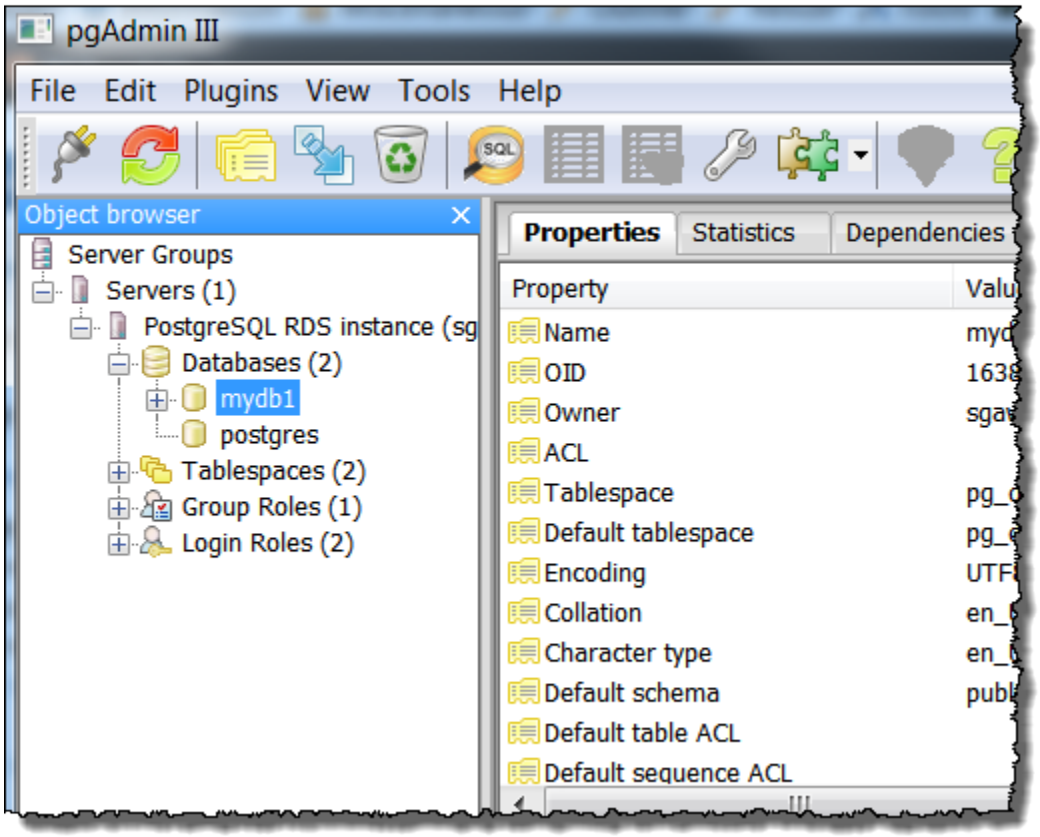
To connect to a PostgreSQL DB instance using pgAdmin

1. Launch the *pgAdmin* application on your client computer. You can install *pgAdmin* from http://www.pgadmin.org/.
2. Select **Add Server** from the **File** menu.
3. In the **New Server Registration** dialog box, enter the DB instance endpoint (for example, mypostgresql.c6c8dntfzzhgv0.us-west-2.rds.amazonaws.com) in the **Host** text box. Do not include the colon or port number as shown on the Amazon RDS console (mypostgresql.c6c8dntfzzhgv0.us-west-2.rds.amazonaws.com:*5432*).

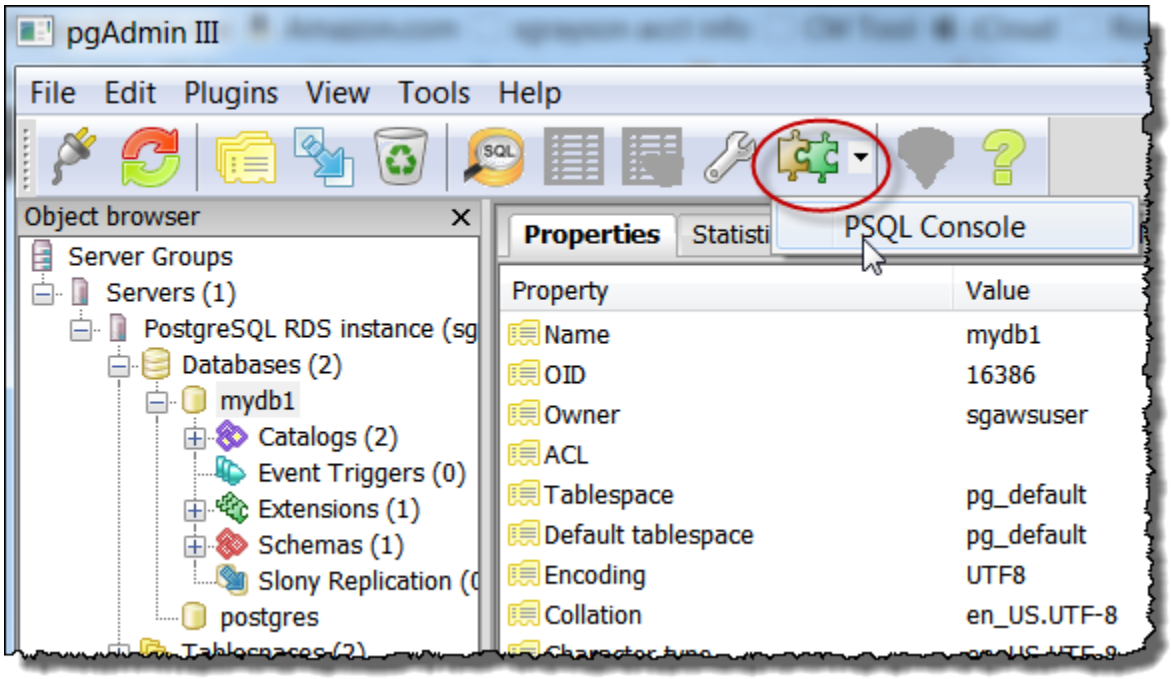
   Enter the port you assigned to the DB instance into the **Port** text box. Enter the user name and user password you entered when you created the DB instance into the **Username** and **Password** text boxes, respectively.



4. Click **OK**.
5. In the **Object browser**, expand the **Server Groups**. Select the Server (the DB instance) you created, and then select the database name.



6. Click the plugin icon and click **PSQL Console**. The *psql* command window opens for the default database you created.



7. Use the command window to enter SQL or *psql* commands. Type `\q` to close the window.

Using *psql* to Connect to a PostgreSQL DB Instance

If your client computer has PostgreSQL installed, you can use a local instance of *psql* to connect to a PostgreSQL DB instance. To connect to your PostgreSQL DB instance using *psql*, you need to provide host information and access credentials.

The following format is used to connect to a PostgreSQL DB instance on Amazon RDS:

```
psql --host=<DB instance endpoint> --port=<port> --username=<master user name>
 --password --dbname=<database name>
```

For example, the following command connects to a database called `mypgdb` on a PostgreSQL DB instance called `mypostgresql` using fictitious credentials:

```
psql --host=mypostgresql.c6c8mwvfdgv0.us-west-2.rds.amazonaws.com --port=5432
--username=awsuser --password --dbname=mypgdb
```

Troubleshooting Connection Issues

By far the most common problem that occurs when attempting to connect to a database on a DB instance is the access rules in the security group assigned to the DB instance. If you used the default DB security group when you created the DB instance, chances are good that the security group did not have the rules that will allow you to access the instance. For more information about Amazon RDS security groups, see Amazon RDS Security Groups (p. 110)

The most common error is *could not connect to server: Connection timed out*. If you receive this error, check that the host name is the DB instance endpoint and that the port number is correct. Check that the security group assigned to the DB instance has the necessary rules to allow access through any firewall your connection may be going through.

Deleting a DB Instance

Once you have connected to the sample DB instance that you created, you should delete the DB instance so you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the **DB Instances** list, select the check box next to the DB instance you wish to delete.
3. Click **Instance Actions**, and then select **Delete** from the dropdown menu.
4. Select **No** in the **Create final Snapshot?** drop-down list box.
5. Click **Yes, Delete**.

Creating a DB Cluster and Connecting to a Database on an Amazon Aurora DB Instance

The easiest way to create an Amazon Aurora DB cluster is to use the Amazon RDS console. Once you have created the DB cluster, you can use standard MySQL utilities such as MySQL Workbench to connect to a database on the DB cluster.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 7) section before you can create or connect to a DB cluster.

Topics

- Create a DB Cluster (p. 47)
- Connect to an Instance in a DB Cluster (p. 52)
- Delete the Sample DB Cluster, DB Subnet Group, and VPC (p. 52)

Create a DB Cluster

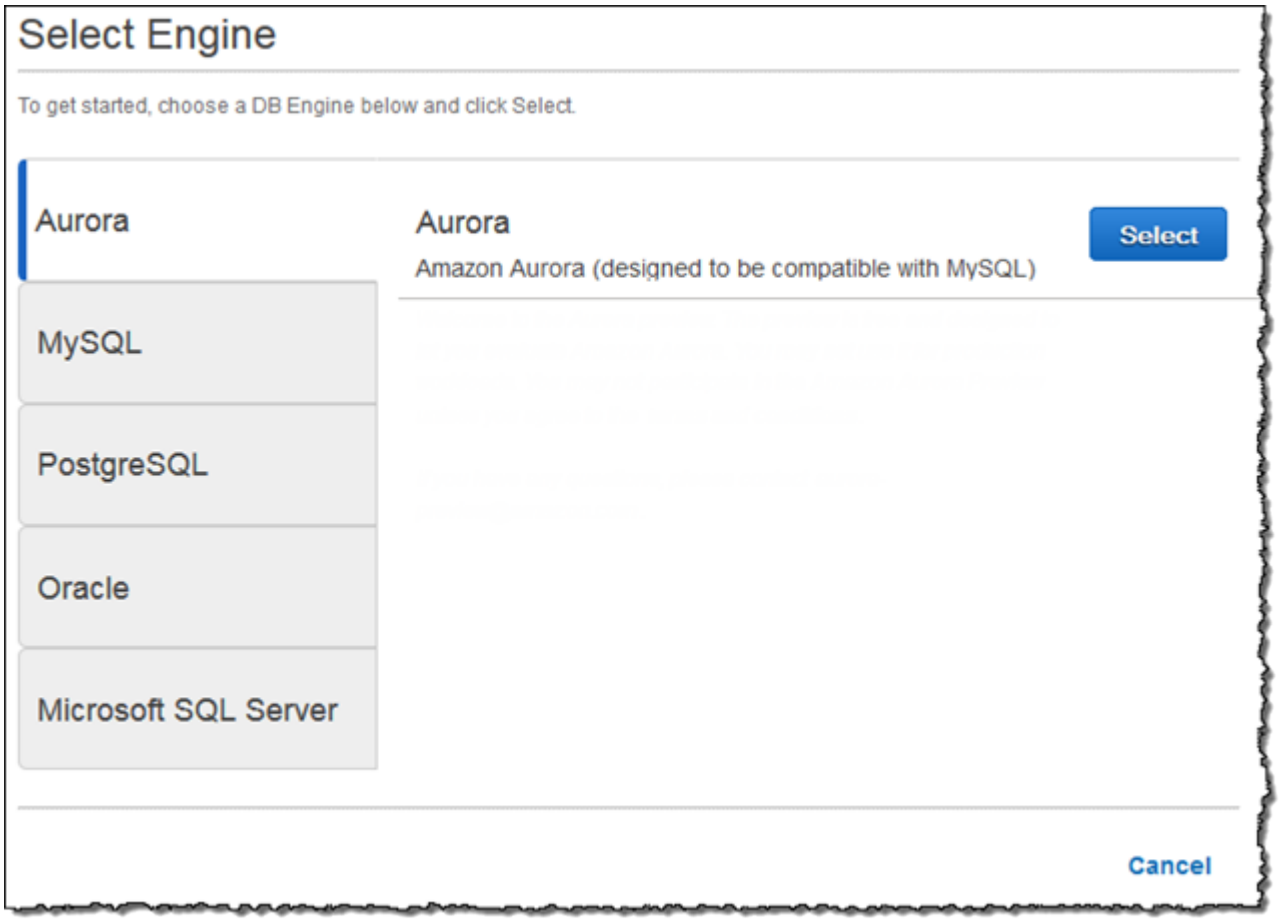
Before you create a DB cluster, you must first have an Amazon Virtual Private Cloud (VPC) and an Amazon RDS DB subnet group. Your VPC must have at least two subnets in at least two Availability Zones. You can use the default VPC for your AWS account, or you can create your own VPC. The Amazon RDS console makes it easy for you to create your own VPC for use with Amazon Aurora or use an existing VPC with your Aurora DB cluster.

If you want to create a VPC and DB subnet group for use with your Amazon Aurora DB cluster yourself, rather than having Amazon RDS create the VPC and DB subnet group for you, then follow the instructions in How to Create a VPC for Use with Amazon Aurora (p. 407). Otherwise, follow the instructions in this topic to create your DB cluster and have Amazon RDS create a VPC and DB subnet group for you.

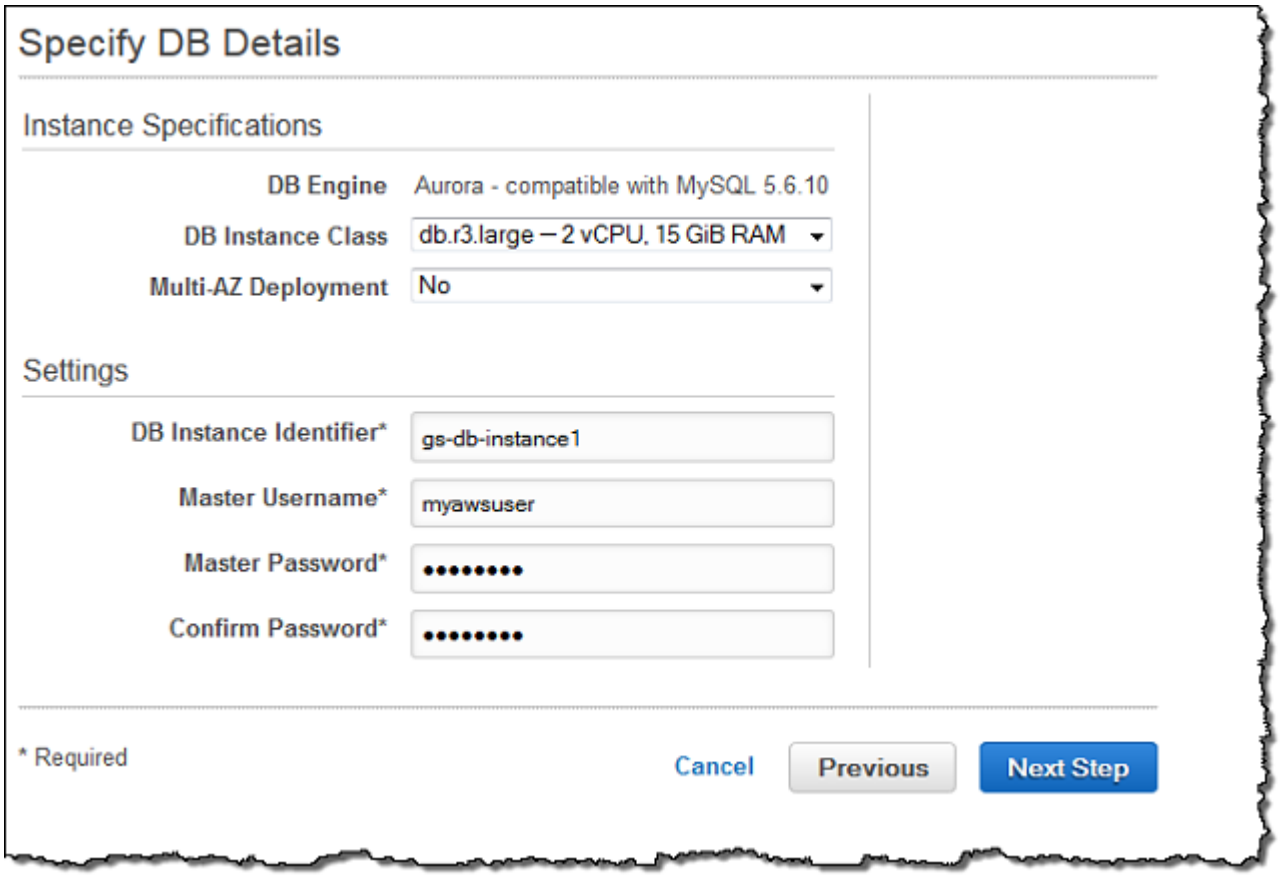
> **Note**
> All VPC and Amazon EC2 resources that you use with your Amazon Aurora DB cluster must and must reside in the US East (N. Virginia), US West (Oregon), or EU (Ireland) regions.

To launch an Aurora DB cluster

1. Open the Amazon RDS for Aurora console at https://console.aws.amazon.com/rds.
2. In the top-right corner of the AWS Management Console, select the region that you want to create your DB cluster in. This example uses the US East (N. Virginia) region. The Aurora preview release is only supported for the US East (N. Virginia), US West (Oregon), or EU (Ireland) regions.
3. In the left navigation pane, click **DB Instances**.
4. Click **Launch DB Instance** to start the Launch DB Instance Wizard. The wizard opens on the Select Engine page.
5. On the Select Engine page, click the **Select** button for the Aurora DB engine.



6. Set the following values on the **Specify DB Details** page:
   - **DB Instance Class:** `db.r3.large`
   - **DB Instance Identifier:** `gs-db-instance1`
   - **Master Username:** Using alphanumeric characters, type a master user name, used to log on to your DB instances in the DB cluster.
   - **Master Password** and **Confirm Password:** Type a password in the **Master Password** box that contains from 8 to 41 printable ASCII characters (excluding /,", and @) for your master user password, used to log on to your database. Then type the password again in the **Confirm Password** box.



7. Click **Next** and set the following values on the **Configure Advanced Settings** page:
   - **VPC ID:** If you have an existing VPC, then you can use that VPC with your Amazon Aurora DB cluster by selecting your VPC identifier, for example `vpc-a464d1c1`. For information on using an existing VPC, see How to Create a VPC for Use with Amazon Aurora (p. 407).

     Otherwise, you can choose to have Amazon RDS create a VPC for you by selecting **Create a new VPC**. This example uses the **Create a new VPC** option.
   - **Subnet Group:** If you have an existing subnet group, then you can use that subnet group with your Amazon Aurora DB cluster by selecting your subnet group identifier, for example, `gs-subnet-group1`.

     Otherwise, you can choose to have Amazon RDS create a subnet group for you by selecting **Create a new subnet group**. This example uses the **Create a new subnet group** option.
   - **Publicly Accessible:** `Yes`

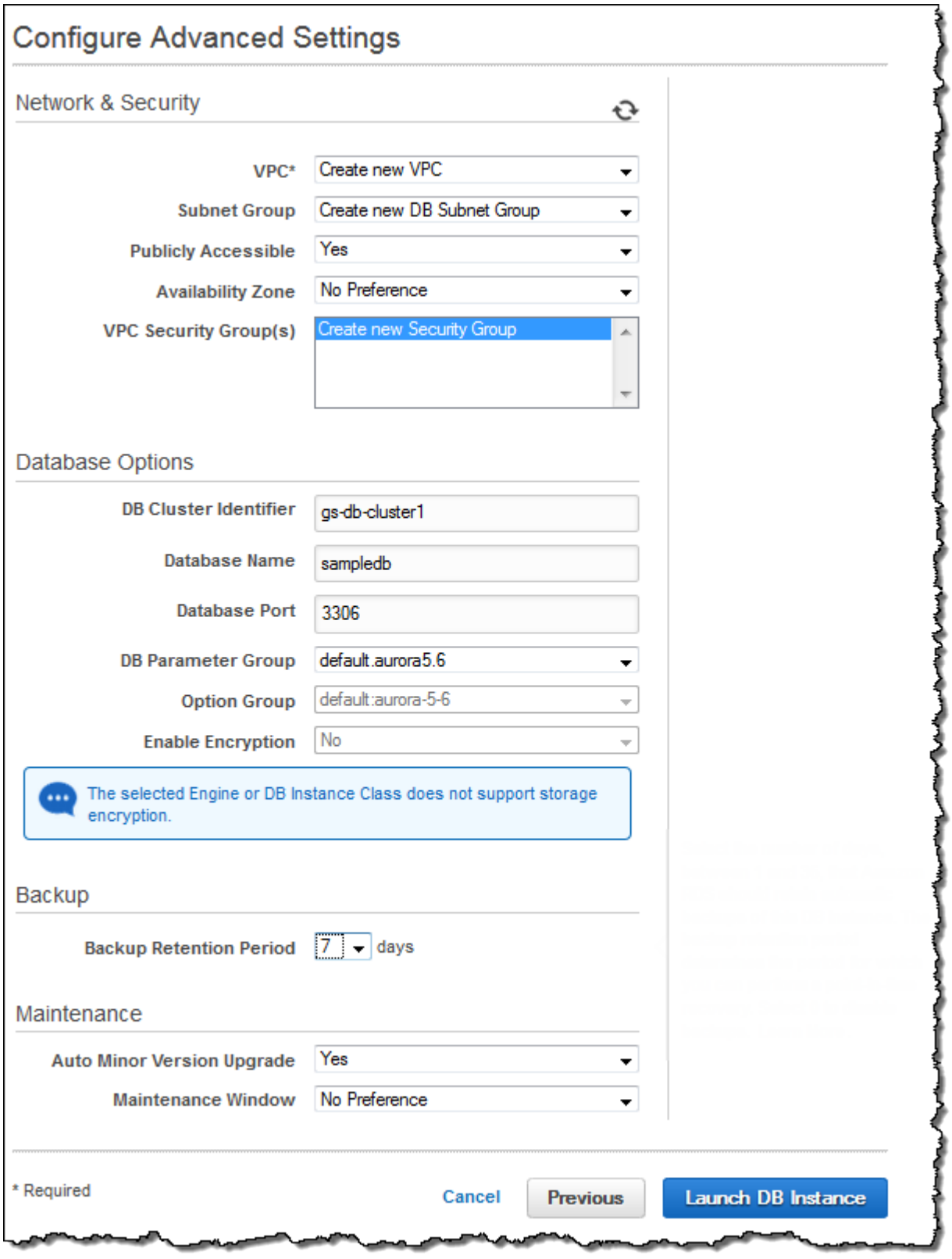
     Note
     Your production DB cluster might not need to be in a public subnet, because only your application servers will require access to your DB cluster. If your DB cluster doesn't need to be in a public subnet, set **Publicly Accessible** to `No`.
   - **Availability Zone:** `No Preference`
   - **VPC Security Group(s):** If you have one or more existing VPC security groups, then you can use one or more of those VPC security groups with your Amazon Aurora DB cluster by selecting your VPC security group identifiers, for example, `gs-security-group1`.

     Otherwise, you can choose to have Amazon RDS create a VPC security group for you by selecting **Create a new Security group**. This example uses the **Create a new Security group** option.

- **DB Cluster Identifier:** `gs-db-cluster1`
- **Database Name:** `sampledb`
- **Database Port:** `3306`

  Note

  You might be behind a corporate firewall that does not allow access to default ports such as the MySQL default port, 3306. In this case, provide a port value that your corporate firewall allows. Remember that port value later when you connect to the Aurora DB cluster.



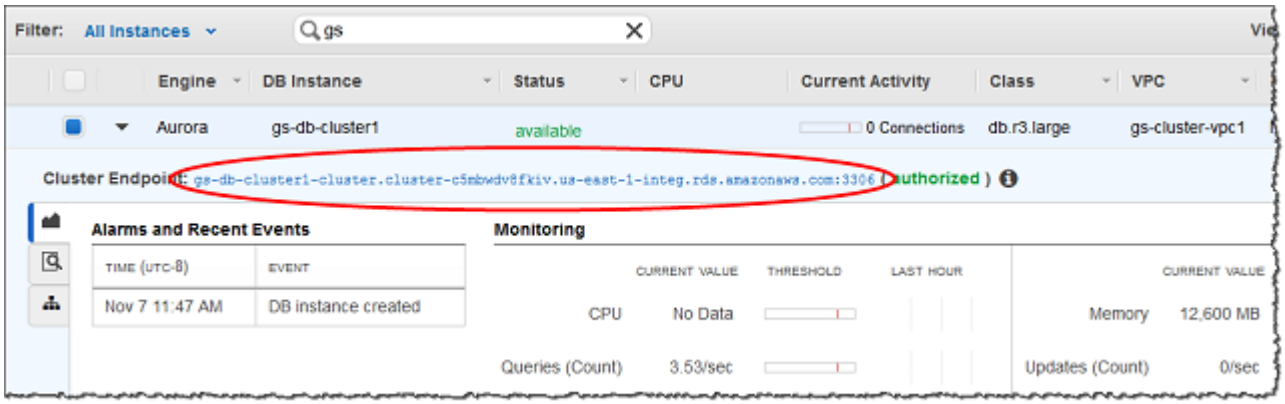
8. Leave the rest of the values as their defaults, and click **Launch DB Instance** to create the DB cluster and primary instance.

Connect to an Instance in a DB Cluster

Once Amazon RDS provisions your DB cluster and creates the primary instance, you can use any standard SQL client application to connect to a database on the DB cluster. In this example, you connect to a database on the DB cluster using MySQL monitor commands. One GUI-based application that you can use to connect is MySQL Workbench. For more information, go to the Download MySQL Workbench page.

To connect to a database on a DB cluster using the MySQL monitor

1. Open the Amazon RDS for Aurora console at https://console.aws.amazon.com/rds.
2. Select **Instances** and click the arrow icon to show the DB cluster details. On the details page, copy the value for the endpoint. This endpoint is the cluster endpoint.



3. Type the following command at a command prompt on a client computer to connect to a database on a DB cluster using the MySQL monitor. Use the cluster endpoint to connect to the primary instance, and the master user name and password that you created previously. If you supplied a port value other than 3306, use that for the `-P` parameter instead.

```
PROMPT> mysql -h <endpoint> -P 3306 -u <mymasteruser> -p <password>
```

You will see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 5.1.32-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Delete the Sample DB Cluster, DB Subnet Group, and VPC

Once you have connected to the sample DB cluster that you created, you can delete the DB cluster, DB subnet group, and VPC (if you created a VPC).

To delete a DB cluster

1. Open the Amazon RDS for Aurora console at https://console.aws.amazon.com/rds.
2. Click **Instances** and then click to select the check box next to the `gs-db-cluster1` DB cluster.
3. Click **Instance Actions**, and then click **Delete** on the dropdown menu.
4. Click **Yes, Delete**.

To delete a DB subnet group

1. Open the Amazon RDS for Aurora console at https://console.aws.amazon.com/rds.
2. Click **Subnet Groups** and then click to select the check box next to the `gs-subnet-group1` DB subnet group.
3. Click **Delete**.
4. Click **Yes, Delete**.

To delete a VPC

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.amazonaws.cn/vpc/.
2. Click **Your VPCs** and then click to select the check box next to the VPC that was created for this procedure.
3. Click **Delete**.
4. Click **Yes, Delete**.

Best Practices for Amazon RDS

This section summarizes best practices for working with Amazon RDS. As new best practices are identified, we will keep this section up to date.

Topics

- Amazon RDS Basic Operational Guidelines (p. 54)
- DB Instance RAM Recommendations (p. 55)
- Amazon RDS Security Best Practices (p. 55)
- Using Metrics to Identify Performance Issues (p. 55)
- Best Practices for Working with MySQL Storage Engines (p. 59)
- Best Practices for Working with PostgreSQL (p. 60)
- Best Practices for Working with SQL Server (p. 62)
- Amazon RDS Best Practices Presentation Video (p. 63)

Amazon RDS Basic Operational Guidelines

The following are basic operational guidelines everyone should follow when working with Amazon RDS. Note that the Amazon RDS Service Level Agreement requires that you follow these guidelines:

- Monitor your memory, CPU, and storage usage. Amazon CloudWatch can be setup to notify you when usage patterns change or when you approach the capacity of your deployment, so that you can maintain system performance and availability.
- Scale up your DB instance when you are approaching storage capacity limits. You should have some buffer in storage and memory to accommodate unforeseen increases in demand from your applications.
- Enable Automatic Backups and set the backup window to occur during the daily low in WriteIOPS.
- On a MySQL DB instance, do not create more than 10,000 tables using Provisioned IOPS or 1000 tables using standard storage. Large numbers of tables will significantly increase database recovery time after a failover or database crash. If you need to create more tables than recommended, set the `innodb_file_per_table` parameter to 0. For more information, see Working with InnoDB Tablespaces to Improve Crash Recovery Times (p. 169) and Working with DB Parameter Groups (p. 523).
- On a MySQL DB instance, avoid tables in your database growing too large. Underlying file system constraints restrict the maximum size of a MySQL table file to 2 TB. Instead, partition your large tables so that file sizes are well under the 2 TB limit. This approach can also improve performance and recovery time. For more information, see MySQL File Size Limits (p. 128).

- If your database workload requires more I/O than you have provisioned, recovery after a failover or database failure will be slow. To increase the I/O capacity of a DB instance, do any or all of the following:
  - Migrate to a DB instance class with High I/O capacity.
  - Convert from standard storage to Provisioned IOPS storage, and use a DB instance class that is optimized for Provisioned IOPS. For information on Provisioned IOPS, see Amazon RDS Provisioned IOPS Storage to Improve Performance (p. 82).
  - If you are already using Provisioned IOPS storage, provision additional throughput capacity.
- If your client application is caching the DNS data of your DB instances, set a TTL of less than 30 seconds. Because the underlying IP address of a DB instance can change after a failover, caching the DNS data for an extended time can lead to connection failures if your application tries to connect to an IP address that no longer is in service.
- Test failover for your DB instance to understand how long the process takes for your use case and to ensure that the application that accesses your DB instance can automatically connect to the new DB instance after failover.

DB Instance RAM Recommendations

An Amazon RDS performance best practice is to allocate enough RAM so that your working set resides almost completely in memory. To tell if your working set is almost all in memory, check the ReadIOPS metric (using AWS CloudWatch) while the DB instance is under load. The value of ReadIOPS should be small and stable. If scaling up the DB instance class---to a class with more RAM---results in a dramatic drop in ReadIOPS, your working set was not almost completely in memory. Continue to scale up until ReadIOPS no longer drops dramatically after a scaling operation, or ReadIOPS is reduced to a very small amount. For information on monitoring a DB instance's metrics, see Viewing DB Instance Metrics (p. 572).

Amazon RDS Security Best Practices

Use AWS IAM accounts to control access to Amazon RDS API actions, especially actions that create, modify, or delete RDS resources such as DB instances, security groups, option groups, or parameter groups, and actions that perform common administrative actions such as backing up and restoring DB instances, or configuring Provisioned IOPS storage.

- Assign an individual IAM account to each person who manages RDS resources. Do not use AWS root credentials to manage Amazon RDS resources; you should create an IAM user for everyone, including yourself.
- Grant each user the minimum set of permissions required to perform his or her duties.
- Use IAM groups to effectively manage permissions for multiple users.
- Rotate your IAM credentials regularly.

For more information about IAM, go to AWS Identity and Access Management. For information on IAM best practices, go to IAM Best Practices.

Using Metrics to Identify Performance Issues

To identify performance issues caused by insufficient resources and other common bottlenecks, you can monitor the metrics available for your Amazon RDS DB instance.

Viewing Performance Metrics

You should monitor performance metrics on a regular basis to see the average, maximum, and minimum values for a variety of time ranges. If you do so, you can identify when performance is degraded. You can also set Amazon CloudWatch alarms for particular metric thresholds so you are alerted if they are reached.

In order to troubleshoot performance issues, it's important to understand the baseline performance of the system. When you set up a new DB instance and get it running with a typical workload, you should capture the average, maximum, and minimum values of all of the performance metrics at a number of different intervals (for example, one hour, 24 hours, one week, two weeks) to get an idea of what is normal. It helps to get comparisons for both peak and off-peak hours of operation. You can then use this information to identify when performance is dropping below standard levels.

To view performance metrics

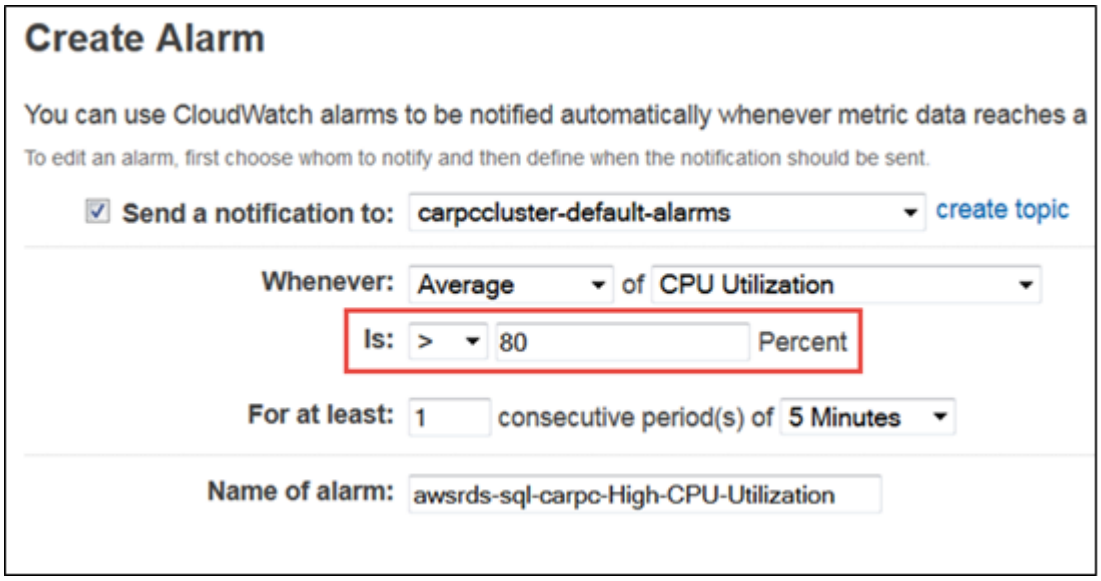
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn//rds/.
2. In the left navigation pane, select **Instances**, and then select a DB instance.
3. Select **Show Monitoring**. The first eight performance metrics display. The metrics default to showing information for the current day.
4. Use the numbered buttons at top right to page through the additional metrics, or select **Show All** to see all metrics.
5. Select a performance metric to adjust the time range in order to see data for other than the current day. You can change the **Statistic**, **Time Range**, and **Period** values to adjust the information displayed. For example, to see the peak values for a metric for each day of the last two weeks, set **Statistic** to **Maximum**, **Time Range** to **Last 2 Weeks**, and **Period** to **Day**.

   Note
   Changing the **Statistic**, **Time Range**, and **Period** values changes them for all metrics. The updated values persist for the remainder of your session or until you change them again.

You can also view performance metrics using the CLI or API. For more information, see Viewing DB Instance Metrics (p. 572).

To set a CloudWatch alarm

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn//rds/.
2. In the left navigation pane, select **Instances**, and then select a DB instance.
3. Select **Show Monitoring**, and then select a performance metric to bring up the expanded view.
4. Select **Create Alarm**.
5. On the **Create Alarm** page, identify what email address should receive the alert by selecting a value in the **Send a notification to** box. Select **create topic** to the right of that box to create a new alarm recipient if necessary.
6. In the **Whenever** list, select the alarm statistic to set.
7. In the **of** box, select the alarm metric.
8. In the **Is** box and the unlabeled box to the right of it, set the alarm threshold, as shown following:



9. In the **For at least** box, enter the number of times that the specified threshold must be reached in order to trigger the alarm.
10. In the **consecutive period(s) of** box, select the period during which the threshold must have been reached in order to trigger the alarm.
11. In the **Name of alarm** box, enter a name for the alarm.
12. Select **Create Alarm**.

The performance metrics page appears, and you can see the new alarm in the **CloudWatch Alarms** status bar. If you don't see the status bar, refresh your page.

Evaluating Performance Metrics

A DB instance has a number of different categories of metrics, and how to determine acceptable values depends on the metric.

Categories of Metrics

CPU

- CPU Utilization – Percentage of computer processing capacity used.

Memory

- Freeable Memory – How much RAM is used by the DB instance, in megabytes.
- Swap Usage – How much swap space is used by the DB instance, in megabytes.

Disk space

- Free Storage Space – How much disk space is used by the DB instance, in megabytes.

Input/output operations

- Read IOPS, Write IOPS – The average number of disk read or write operations per second.
- Read Latency, Write Latency – The average time for a read or write operation in milliseconds.
- Read Throughput, Write Throughput – The average number of megabytes read from or written to disk per second.
- Queue Depth – The number of I/O operations that are waiting to be written to or read from disk.

Network traffic

- Network Receive Throughput, Network Transmit Throughput – The rate of network traffic to and from the DB instance in megabytes per second.

Database connections

- DB Connections – The number of client sessions that are connected to the DB instance.

For more detailed individual descriptions of the performance metrics available, see Amazon RDS Dimensions and Metrics. For an idea of the acceptable values for metrics, see Acceptable Values for Metrics.

Acceptable Values for Metrics

Generally speaking, acceptable values for performance metrics depend on what your baseline looks like and what your application is doing. Investigate consistent or trending variances from your baseline. Advice about specific types of metrics follows:

- **High CPU or RAM consumption –** High values for CPU or RAM consumption might be appropriate, provided that they are in keeping with your goals for your application (like throughput or concurrency) and are expected.
- **Disk space consumption –** Investigate disk space consumption if space used is consistently at or above 85 percent of the total disk space. See if it is possible to delete data from the instance or archive data to a different system to free up space.
- **Network traffic –** For network traffic, talk with your system administrator to understand what expected throughput is for your domain network and Internet connection. Investigate network traffic if throughput is consistently lower than expected.
- **Database connections –** Consider constraining database connections if you see high numbers of user connections in conjunction with decreases in instance performance and response time. The best number of user connections for your DB instance will vary based on your instance class and the complexity of the operations being performed. You can determine the number of database connections by associating your DB instance with a parameter group where the *User Connections* parameter is set to other than 0 (unlimited). You can either use an existing parameter group or create a new one. For more information, see Working with DB Parameter Groups (p. 523).
- **IOPS metrics –** The expected values for IOPS metrics depend on disk specification and server configuration, so use your baseline to know what is typical. Investigate if values are consistently different than your baseline. For best IOPS performance, make sure your typical working set will fit into memory to minimize read and write operations.

For issues with any performance metrics, one of the first things you can do to improve performance is tune the most used and most expensive queries to see if that lowers the pressure on system resources. For more information, see Tuning Queries (p. 59)

If your queries are tuned and an issue persists, consider upgrading your Amazon RDS DB Instance Class (p. 65) to one with more of the resource (CPU, RAM, disk space, network bandwidth, I/O capacity) that is related to the issue you are experiencing.

Tuning Queries

One of the best ways to improve DB instance performance is to tune your most commonly used and most resource-intensive queries to make them less expensive to run.

MySQL Query Tuning

Go to Optimizing SELECT Statements in the MySQL documentation for more information on writing queries for better performance. You can also go to MySQL Performance Tuning and Optimization Resources for additional query tuning resources.

Oracle Query Tuning

Go to the Database SQL Tuning Guide in the Oracle documentation for more information on writing and analyzing queries for better performance.

SQL Server Query Tuning

Go to Analyzing a Query in the SQL Server documentation to improve queries for SQL Server DB instances. You can also use the execution-, index- and I/O-related data management views (DMVs) described in the Dynamic Management Views and Functions documentation to troubleshoot SQL Server query issues.

A common aspect of query tuning is creating effective indexes. You can use the Database Engine Tuning Advisor to get potential index improvements for your DB instance. For more information, see Analyzing Your Database Workload on a DB Instance Using SQL Server Tuning Advisor (p. 344).

PostgreSQL Query Tuning

Go to Using EXPLAIN in the PostgreSQL documentation to learn how to analyze a query plan. You can use this information to modify a query or underlying tables in order to improve query performance. You can also go to Controlling the Planner with Explicit JOIN Clauses to get tips about how to specify joins in your query for the best performance.

Best Practices for Working with MySQL Storage Engines

The Point-In-Time-Restore and snapshot restore features of Amazon RDS for MySQL require a crash recoverable storage engine and are supported for the InnoDB storage engine only. While MySQL supports multiple storage engines with varying capabilities, not all of them are optimized for crash recovery and data durability. For example, the MyISAM storage engine does not support reliable crash recovery and might prevent a Point-In-Time restore or snapshot restore from working as intended. This may result in lost or corrupt data when MySQL is restarted after a crash.

InnoDB is the recommended and supported storage engine for MySQL DB instances on Amazon RDS. However, MyISAM performs better than InnoDB if you require intense, full-text search capability. If you still choose to use MyISAM with Amazon RDS, following these steps outlined in Automated Backups with Unsupported MySQL Storage Engines (p. 75) can be helpful in certain scenarios for snapshot restore functionality.

If you would like to convert existing MyISAM tables to InnoDB tables, you can use the process outlined in the MySQL documentation. MyISAM and InnoDB have different strengths and weaknesses, so you should fully evaluate the impact of making this switch on your applications before doing so.

In addition, Federated Storage Engine is currently not supported by Amazon RDS for MySQL.

Best Practices for Working with PostgreSQL

Two important areas where you can improve performance with PostgreSQL on Amazon RDS are when loading data into a DB instance and when using the PostgreSQL autovacuum feature. The following sections cover some of the practices we recommend for these areas.

Loading Data into a PostgreSQL DB Instance

When loading data into an Amazon RDS PostgreSQL DB instance, you should modify your DB instance settings and your DB parameter group values to allow for the most efficient importing of data into your DB instance.

Modify your DB instance settings to the following:

- Disable DB instance backups (set backup_retention to 0)
- Disable Multi-AZ

Modify your DB parameter group to include the following settings. You should test the parameter settings to find the most efficient settings for your DB instance:

- Increase the value of the `maintenance_work_mem` parameter. For more information about PostgreSQL resource consumption parameters, see the PostgreSQL documentation.
- Increase the value of the `checkpoint_segments` and `checkpoint_timeout` parameters to reduce the number of writes to the wal log.
- Disable the `synchronous_commit` parameter (do not turn off FSYNC).
- Disable the PostgreSQL autovacuum parameter.

Use the `pg_dump -Fc` (compressed) or `pg_restore -j` (parallel) commands with these settings.

Working with the fsync and full_page_writes database parameters

In PostgreSQL 9.4.1 on Amazon RDS, the `fsync` and `full_page_writes` database parameters are not modifiable. Disabling the `fsync` and `full_page_writes` database parameters can lead to data corruption, so we have enabled them for you. We recommend that customers with other 9.3 DB engine versions of PostgreSQL not disable the `fsync` and `full_page_writes` parameters.

Working with the PostgreSQL Autovacuum Feature

The autovacuum feature for PostgreSQL databases is a feature that we strongly recommend you use to maintain the health of your PostgreSQL DB instance. Autovacuum automates the execution of the VACUUM and ANALYZE command; using autovacuum is required by PostgreSQL, not imposed by Amazon RDS, and its use is critical to good performance. The feature is enabled by default for all new Amazon RDS PostgreSQL DB instances, and the related configuration parameters are appropriately set by default.

Your database administrator needs to know and understand this maintenance operation. For the PostgreSQL documentation on autovacuum, see http://www.postgresql.org/docs/current/static/routine-vacuuming.html#AUTOVACUUM.

Autovacuum is not a “resource free” operation, but it works in the background and yields to user operations as much as possible. When enabled, autovacuum checks for tables that have had a large number of updated or deleted tuples. It also protects against loss of very old data due to transaction ID wraparound.

Autovacuum should not be thought of as a high-overhead operation that can be reduced to gain better performance. On the contrary, tables that have a high velocity of updates and deletes will quickly deteriorate over time if autovacuum is not run.

Important
Not running autovacuum can result in an eventual required outage to perform a much more intrusive vacuum operation. When an Amazon RDS PostgreSQL DB instance becomes unavailable because of an over conservative use of autovacuum, the PostgreSQL database will shut down to protect itself. At that point, Amazon RDS must perform a single-user–mode full vacuum directly on the DB instance , which can result in a multi-hour outage. Thus, we strongly recommend that you do not turn off autovacuum, which is enabled by default.

The autovacuum parameters determine when and how hard autovacuum works. The `autovacuum_vacuum_threshold` and `autovacuum_vacuum_scale_factor` parameters determine when autovacuum is run. The `autovacuum_max_workers`, `autovacuum_nap_time`, `autovacuum_cost_limit`, and `autovacuum_cost_delay` parameters determine how hard autovacuum works. For more information about autovacuum, when it runs, and what parameters are required, see the PostgreSQL documentation.

The following query shows the number of "dead" tuples in a table named table1 :

```
PROMPT> select relname, n_dead_tup, last_vacuum, last_autovacuum from
pg_catalog.pg_stat_all_tables
where n_dead_tup > 0 and relname =  'table1' order by n_dead_tup desc;
```

The results of the query will resemble the following:

```
relname | n_dead_tup | last_vacuum | last_autovacuum
--------+------------+-------------+-----------------
 tasks   |   81430522 |             |
(1 row)
```

The following query shows what vacuum-related processes are currently active:

```
tasks=#  select pid, usename, waiting, state, query from pg_stat_activity where
 query like '%vacuum%';
```

The results of the query will resemble the following:

```
pid  | usename  | waiting | state  |
```

```
   query
-------+----------+---------+--------+-----------------------------------
-------------------------------------------------------
 26784 | rdsadmin | f       | active | autovacuum: VACUUM public.tasks (to
prevent wraparound)
 24709 | rdsadmin | f       | active | select pid, usename, waiting, state,
query from pg_stat_activity where query like '%vacuum%';
(2 rows)
```

Best Practices for Working with SQL Server

Best practices for a Multi-AZ deployment with a SQL Server DB instance include the following:

- Use Amazon RDS DB events to monitor failovers. For example, you can be notified by text message or email when a DB instance fails over. For more information about Amazon RDS events, see Using Amazon RDS Event Notification (p. 575).
- If your application caches DNS values, set time to live (TTL) to less than 30 seconds. Setting TTL as so is a good practice in case there is a failover, where the IP address might change and the cached value might no longer be in service.
- We recommend that you *do not* enable the following modes because they turn off transaction logging, which is required for Multi-AZ:
  - Simple recover mode
  - Offline mode
  - Read-only mode
- Test to determine how long it takes for your DB instance to failover. Failover time can vary due to the type of database, the instance class, and the storage type you use. You should also test your application's ability to continue working if a failover occurs.
- To shorten failover time, you should do the following:
  - Ensure that you have sufficient Provisioned IOPS allocated for your workload. Inadequate I/O can lengthen failover times. Database recovery requires I/O.
  - Use smaller transactions. Database recovery relies on transactions, so if you can break up large transactions into multiple smaller transactions, your failover time should be shorter.
- Take into consideration that during a failover, there will be elevated latencies. As part of the failover process, Amazon RDS automatically replicates your data to a new standby instance. This replication means that new data is being committed to two different DB instances, so there might be some latency until the standby DB instance has caught up to the new primary DB instance.
- Deploy your applications in all Availability Zones. If an Availability Zone does go down, your applications in the other Availability Zones will still be available.

When working with a Multi-AZ deployment of SQL Server, remember that Amazon RDS mirrors all SQL Server databases on your instance. If you don't want particular databases to be mirrored, set up a separate DB instance that doesn't use Multi-AZ for those databases.

Amazon RDS SQL Server Best Practices Video

The 2014 AWS re:Invent conference included a presentation on best practices for SQL Server on Amazon RDS. A video of the presentation is available here.

Amazon RDS Best Practices Presentation Video

The 2013 AWS re:Invent conference included a presentation on best practices for performance-intensive, production applications. A video of the presentation is available here.

Amazon RDS DB Instances

A *DB instance* is an isolated database environment running in the cloud. It is the basic building block of Amazon RDS. A DB instance can contain multiple user-created databases, and can be accessed using the same client tools and applications you might use to access a stand-alone database instance. DB instances are simple to create and modify with the Amazon RDS command line tools, APIs, or the AWS Management RDS Console.

> **Note**
> Amazon RDS supports access to databases using any standard SQL client application. Amazon RDS does not allow direct host access.

You can have up to 40 Amazon RDS DB instances. Of these 40, up to 10 can be Oracle or SQL Server DB instances under the "License Included" model. All 40 DB instances can be used for MySQL or PostgreSQL. You can also have 40 DB instances for SQL Server or Oracle under the "BYOL" licensing model. If your application requires more DB instances, you can request additional DB instances using the form at https://console.amazonaws.cn/support/home#/case/create?issueType=service-limit-increase&limitType=service-code-rds-instances.

Each DB instance has a DB instance identifier. This customer-supplied name uniquely identifies the DB instance when interacting with the Amazon RDS API and commands. The DB instance identifier must be unique for that customer in an AWS region.

Each DB instance supports a database engine. Amazon RDS currently supports MySQL, PostgreSQL, Oracle, and Microsoft SQL Server database engines.

Amazon RDS has released a new DB engine: Aurora. The Amazon Aurora DB engine supports multiple DB instances in a DB cluster. Amazon Aurora is currently in preview release and is subject to change. For detailed information, see Aurora on Amazon RDS (p. 392).

When creating a DB instance, some database engines require that a database name be specified. A DB instance can host multiple databases, or a single Oracle database with multiple schemas. The database name value depends on the database engine:

- For the MySQL database engine, the database name is the name of a database hosted in your DB instance. Databases hosted by the same DB instance must have a unique name within that instance.
- For the Oracle database engine, database name is used to set the value of ORACLE_SID, which must be supplied when connecting to the Oracle RDS instance.
- For the Microsoft SQL Server database engine, database name is not a supported parameter.

- For the PostgreSQL database engine, the database name is the name of a database hosted in your DB instance. A database name is not required when creating a DB instance. Databases hosted by the same DB instance must have a unique name within that instance.

Amazon RDS creates a master user account for your DB instance as part of the creation process. This master user has permissions to create databases and to perform create, delete, select, update and insert operations on tables the master user creates. You must set the master user password when you create a DB instance, but you can change it at any time using the Amazon RDS command line tools, APIs, or the AWS Management Console. You can also change the master user password and manage users using standard SQL commands.

Topics

- DB Instance Class (p. 65)
- DB Instance Status (p. 68)
- Regions and Availability Zones (p. 69)
- High Availability (Multi-AZ) (p. 71)
- Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73)
- DB Instance Backups (p. 73)
- DB Instance Replication (p. 76)

DB Instance Class

The computation and memory capacity of a DB instance is determined by its DB instance class. You can change the CPU and memory available to a DB instance by changing its DB instance class; to change the DB instance class, you must modify the DB instance. For pricing information on DB instance classes, go to the Amazon RDS pricing page.

The following list describes the Amazon RDS DB instance class types and the Amazon EC2 instance type it uses:

- **Micro Instances (db.t1.micro)** – An instance sufficient for testing that should not be used for production applications. Using a *db.t1.micro* instance with Oracle is a limited test configuration. We recommend you use *db.t1.micro* instances with Oracle to test setup and connectivity only; the system resources for a *db.t1.micro* instance do not meet the recommended configuration for Oracle. No Oracle options are supported on a db.t1.micro instance. For more information, see the Micro Instances topic in the Amazon EC2 documentation.
- **Standard – Current Generation (db.m3)** – Second generation instances that provide more computing capacity than the first generation *db.m1* instance classes at a lower price.
- **Memory Optimized – Current Generation (db.r3)** – Second generation instances that provide memory optimization and more computing capacity than the first generation *db.m2* instance classes at a lower price. AWS provides *db.r3* DB instance classes for MySQL 5.6, PostgreSQL, and SQL Server DB instances. The *db.r3* DB instances classes are not available for Oracle instances, and are not available in the South America (Sao Paulo), China (Beijing), and AWS GovCloud (US) regions.

  Some SQL Server editions have limitations with *db.r3* instances classes. SQL Server Express is not supported with *db.r3* instance classes due to Microsoft licensing restrictions. SQL Server Standard and SQL Server Web are limited to *db.r3.2xlarge* and smaller DB instance classes due to the editions' memory and CPU limitations. Currently, SQL Server Multi-AZ deployments using *db.r3* instances classes are only available for SQL Server Standard and SQL Server Enterprise.

  MySQL DB instances created after April 23, 2014, can switch to the *db.r3* instance classes by modifying the DB instance just as with any other modification. MySQL DB instances running MySQL versions 5.1 or 5.5 and created before April 23, 2014 must first upgrade to MySQL version 5.6 For information on

upgrading a MySQL DB instance, see Upgrading Database Versions for a DB Instance (p. 446). For more information, go to R3 Instances in the Amazon EC2 documentation.

- **Burst Capable – Current Generation (db.t2)** – Instances that provide baseline performance level with the ability to burst to full CPU usage. This DB instance class requires that the DB instance be in a VPC. Currently, it is not supported in AWS GovCloud (US) and for Multi-AZ deployments in South America (Sao Paulo). If you have an existing DB instance that you want to move to the *db.t2* DB instance class, note that the *db.t2* DB instance class requires a VPC; if your current DB instance is not in a VPC, see Moving a DB Instance not in a VPC into a VPC (p. 568) to find out how to move a DB instance not in a VPC into a VPC. For more information about T2 instances used with the *db.t2* DB instance class, go to T2 Instances in the Amazon EC2 documentation.

  Burst capable instances (db.t2) are available for the following DB engines:

DB Engine	Availability
MySQL	MySQL version 5.6 supported
Oracle	Not supported
SQL Server	• SQL Server Express supports *db.t2.micro*, *db.t2.small*, and *db.t2.medium*. • SQL Server Standard with Bring Your Own License (BYOL) supports *db.t2.small* and *db.t2.medium*. SQL Server Standard with License Included (LI) is not supported. • SQL Server Web supports *db.t2.small* and *db.t2.medium*. • SQL Server Enterprise with Bring Your Own License (BYOL) supports *db.t2.small* and *db.t2.medium*. • SQL Server Multi-AZ deployments using SQL Server mirroring are supported for SQL Server Standard and SQL Server Enterprise in three regions: US East (N. Virginia), US West (Oregon), EU (Ireland).
PostgreSQL	All versions supported
Aurora	Not supported

  If you have a DB instance that is supported for *db.t2* but that is not in a VPC, you must move the instance into a VPC before you can convert to the *db.t2* instance class. For information on moving a DB instance into a VPC, see Moving a DB Instance not in a VPC into a VPC (p. 568)
- **High Memory – Previous Generation (db.cr1)** – Instances that are only supported for MySQL 5.6 and PostgreSQL DB instances and are available in the US East (N. Virginia), US West (Oregon), EU (Ireland), and Asia Pacific (Tokyo) regions. This DB instance class, when used with MySQL 5.6 or PostgreSQL and Provisioned IOPS, can realize up to 20,000 IOPS for MySQL and 25,000 IOPS for PostgreSQL.
- **Memory Optimized – Previous Generation (db.m2**) – First generation memory-optimized instances. For more information, go to Instance Type in the Amazon EC2 documentation.
- **Standard – Previous Generation (db.m1)** – First generation standard instances. For more information, go to Instance Type in the Amazon EC2 documentation.

The following table provides details of the Amazon RDS DB instance classes.

Instance Class	vCPU	ECU	Memory (GB)	EBS Optimized	Network Performance
Micro Instances					
db.t1.micro	1	1	.615	No	Very Low

Instance Class	vCPU	ECU	Memory (GB)	EBS Optimized	Network Performance
db.m1.small	1	1	1.7	No	Very Low
Standard - Current Generation					
db.m3.medium	1	3	3.75	No	Moderate
db.m3.large	2	6.5	7.5	No	Moderate
db.m3.xlarge	4	13	15	500 Mbps	High
db.m3.2xlarge	8	26	30	1000 Mbps	High
Memory Optimized - Current Generation					
db.r3.large	2	6.5	15	No	Moderate
db.r3.xlarge	4	13	30.5	500 Mbps	Moderate
db.r3.2xlarge	8	26	61	1000 Mbps	High
db.r3.4xlarge	16	52	122	2000 Mbps	High
db.r3.8xlarge	32	104	244	No	10 Gbps
Burst Capable - Current Generation					
db.t2.micro	1	1	1	No	Low
db.t2.small	1	1	2	No	Low
db.t2.medium	2	2	4	No	Moderate
Memory Optimized - Previous Generation					
db.m2.xlarge	2	6.5	17.1	No	Moderate
db.m2.2xlarge	4	13	34.2	500 Mbps	Moderate
db.m2.4xlarge	8	26	68.4	1000 Mbps	High
db.cr1.8xlarge	32	88	244	No	10 Gbps
Standard - Previous Generation					
db.m1.medium	1	2	3.75	No	Moderate
db.m1.large	2	4	7.5	500 Mbps	Moderate
db.m1.xlarge	4	8	15	1000 Mbps	High

Note
The table column information includes:

- **vCPU** – A virtual CPU, or virtual central processing unit, is a unit of capacity that you can use to compare DB instance classes. Instead of purchasing or leasing a particular processor to

use for several months or years, you are renting capacity by the hour. Our goal is to provide a consistent amount of CPU capacity no matter what the actual underlying hardware.

- **ECU** – The EC2 Compute Unit provides the relative measure of the integer processing power of an Amazon EC2 instance. In order to make it easy for developers to compare CPU capacity between different instance classes, we have defined an Amazon EC2 Compute Unit. The amount of CPU that is allocated to a particular instance is expressed in terms of these EC2 Compute Units. One ECU currently provides CPU capacity equivalent to a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.
- **Memory (GB)** – Specifies the RAM memory, in gigabytes, allocated to the DB instance. Note that there is often a consistent ratio between memory and vCPU. For example, the *db.m1* DB instance class has the same memory to vCPU ratio as the *db.m3* DB instance class, but *db.m3* instance classes provide better, more consistent performance that *db.m1* instances for most use cases. *db.m3* instance classes are also less expensive than *db.m1* instances.
- **EBS-optimized** – DB instance uses an optimized configuration stack and provides additional, dedicated capacity for Amazon Elastic Block Store (Amazon EBS) I/O. This optimization provides the best performance for your Amazon EBS volumes by minimizing contention between Amazon EBS I/O and other traffic from your instance. For more information about Amazon EBS–optimized instances, go to Amazon EBS–Optimized Instances in the Amazon EC2 documentation.
- **Network Performance** – The network speed relative to other DB instance classes.

DB Instance Status

The status of a DB instance indicates the health of the instance. You can view the status of a DB instance by using the RDS console, the CLI command `rds-describe-db-instances`, or the API action `DescribeDBInstances`.

Note
Amazon RDS also uses another status called *maintenance status*, which is shown in the Maintenance column of the Amazon RDS console. This value indicates the status of any maintenance patches that need to be applied to a DB instance. Maintenance status is independent of DB instance status. For more information on *maintenance status*, see Upgrading and Amazon RDS Resource Maintenance (p. 442).

DB Instance Status	Description
available	The instance is healthy and available.
backing-up	The instance is currently being backed up.
creating	The instance is being created. The instance is inaccessible while it is being created.
deleting	The instance is being deleted.
failed	The instance has failed and Amazon RDS was unable to recover it. Perform a point-in-time restore to the latest restorable time of the instance to recover the data.
inaccessible-encryption-credentials	The KMS key used to encrypt or decrypt the DB instance could not be accessed.
incompatible-credentials	The supplied CloudHSM username or password is incorrect. Please update the CloudHSM credentials for the DB instance.

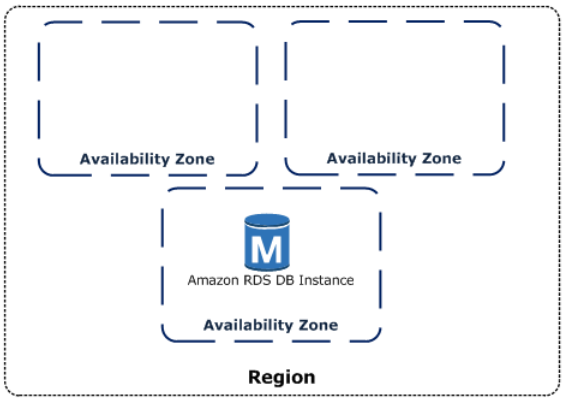
DB Instance Status	Description
incompatible-network	Amazon RDS is attempting to perform a recovery action on an instance but is unable to do so because the VPC is in a state that is preventing the action from being completed. This status can occur if, for example, all available IP addresses in a subnet were in use and Amazon RDS was unable to get an IP address for the DB instance.
incompatible-option-group	Amazon RDS attempted to apply an option group change but was unable to do so, and Amazon RDS was unable to roll back to the previous option group state. Consult the Recent Events list for the DB instance for more information. This status can occur if, for example, the option group contains an option such as TDE and the DB instance does not contain encrypted information.
incompatible-parameters	Amazon RDS was unable to start up the DB instance because the parameters specified in the instance's DB parameter group were not compatible. Revert the parameter changes or make them compatible with the instance to regain access to your instance. Consult the Recent Events list for the DB instance for more information about the incompatible parameters.
incompatible-restore	Amazon RDS is unable to do a point-in-time restore. Common causes for this status include using temp tables or using MyISAM tables.
maintenance	Amazon RDS is applying a maintenance update to the DB instance.
modifying	The instance is being modified because of a customer request to modify the instance.
rebooting	The instance is being rebooted because of a customer request or an Amazon RDS process that requires the rebooting of the instance.
renaming	The instance is being renamed because of a customer request to rename it.
resetting-master-credentials	The master credentials for the instance are being reset because of a customer request to reset them.
restore-error	The DB instance encountered an error attempting to restore to a point-in-time or from a snapshot.
storage-full	The instance has reached its storage capacity allocation. This is a critical status and should be remedied immediately; you should scale up your storage by modifying the DB instance. Set CloudWatch alarms to warn you when storage space is getting low so you don't run into this situation.
upgrading	The database engine version is being upgraded.

Regions and Availability Zones

Amazon cloud computing resources are housed in highly available data center facilities in different areas of the world (for example, North America, Europe, and Asia). Each data center location is called a region.

Each region contains multiple distinct locations called Availability Zones, or AZs. Each Availability Zone is engineered to be isolated from failures in other Availability Zones, and to provide inexpensive, low-latency

network connectivity to other zones in the same region. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location.



It is important to remember that each region is completely independent. Any Amazon RDS activity you initiate (for example, creating database instances or listing available database instances) runs only in your current default region. The default region can be changed in the console, by setting the EC2_REGION environment variable, or it can be overridden by using the `--url` parameter with the command line interface. See *Common Options for API Tools* for more information.

Amazon RDS supports a special AWS region called AWS GovCloud (US) that is designed to allow US government agencies and customers to move more sensitive workloads into the cloud by addressing their specific regulatory and compliance requirements. For more information on AWS GovCloud (US), see the AWS GovCloud (US) home page.

To create or work with an Amazon RDS DB instance in a specific region, use the corresponding regional service endpoint.

Amazon RDS supports the endpoints listed in the following table.

Region	Name	Endpoint
US East (N. Virginia) region	us-east-1	https://rds.us-east-1.amazonaws.com
US West (N. California) region	us-west-1	https://rds.us-west-1.amazonaws.com
US West (Oregon) region	us-west-2	https://rds.us-west-2.amazonaws.com
EU (Ireland) region	eu-west-1	https://rds.eu-west-1.amazonaws.com
EU (Frankfurt) Region	eu-central-1	https://rds.eu-central-1.amazonaws.com
Asia Pacific (Tokyo) Region	ap-northeast-1	https://rds.ap-northeast-1.amazonaws.com
Asia Pacific (Singapore) Region	ap-southeast-1	https://rds.ap-southeast-1.amazonaws.com
Asia Pacific (Sydney) Region	ap-southeast-2	https://rds.ap-southeast-2.amazonaws.com

Region	Name	Endpoint
South America (Sao Paulo) Region	sa-east-1	https://rds.sa-east-1.amazonaws.com
China (Beijing) Region	cn-north-1	https://rds.cn-north-1.amazonaws.com
AWS GovCloud (US) Region	us-gov-west-1	https://rds.us-gov-west-1.amazonaws.com

If you do not explicitly specify an endpoint, the US West (Oregon) endpoint is the default.

Related Topics

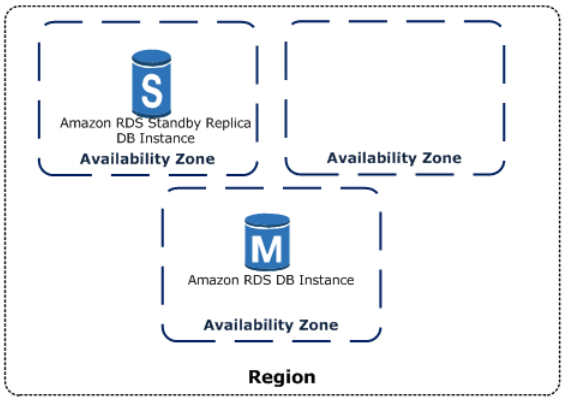
- Regions and Availability Zones in the *Amazon Elastic Compute Cloud User Guide*.
- Amazon RDS DB Instances (p. 64)

High Availability (Multi-AZ)

Amazon RDS provides high availability and failover support for DB instances using Multi-AZ deployments. Multi-AZ deployments for Oracle, PostgreSQL, and MySQL DB instances use Amazon technology, while SQL Server DB instances use SQL Server Mirroring. In a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. The primary DB instance is synchronously replicated across Availability Zones to a standby replica to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups. Running a DB instance with high availability can enhance availability during planned system maintenance, and help protect your databases against DB instance failure and Availability Zone disruption. For more information on Availability Zones, see Regions and Availability Zones (p. 69).

> **Note**
> The high-availability feature is not a scaling solution for read-only scenarios; you cannot use a standby replica to serve read traffic. To service read-only traffic, you should use a Read Replica. For more information, see Working with PostgreSQL and MySQL Read Replicas (p. 472).

When using the BYOL licensing model, you must have a license for both the primary instance and the standby replica.



Using the RDS console, you can create a Multi-AZ deployment by simply specifying Multi-AZ when creating a DB instance. You can also use the console to convert existing DB instances to Multi-AZ deployments by modifying the DB instance and specifying the Multi-AZ option. The RDS console shows the Availability Zone of the standby replica, called the secondary AZ.

You can specify a Multi-AZ deployment using the CLI as well. For SQL Server Multi-AZ deployments using SQL Server Mirroring, you specify the option in an option group; for more information on the SQL Server option for Mirroring, see Multi-AZ Deployment for SQL Server Using the Mirroring Option (p. 353). Use the RDS CLI command rds-describe-db-instances or the API action DescribeDBInstances to show the Availability Zone of the standby replica (called the secondary AZ).

The RDS console shows the Availability Zone of the standby replica (called the secondary AZ), or you can use the command rds-describe-db-instances or the API action DescribeDBInstances to find the secondary AZ. When using the BYOL licensing model, you must have a license for both the primary instance and the standby replica.

DB instances using Multi-AZ deployments may have increased write and commit latency compared to a Single-AZ deployment, due to the synchronous data replication that occurs. You may have a change in latency if your deployment fails over to the standby replica, although AWS is engineered with low-latency network connectivity between Availability Zones. For production workloads, we recommend you use Provisioned IOPS and DB instance classes (m1.large and larger) that are optimized for Provisioned IOPS for fast, consistent performance.

Failover Process for Amazon RDS

In the event of a planned or unplanned outage of your DB instance, Amazon RDS automatically switches to a standby replica in another Availability Zone if you have enabled Multi-AZ. The time it takes for the failover to complete depends on the database activity and other conditions at the time the primary DB instance became unavailable. Failover times are typically 60-120 seconds. However, large transactions or a lengthy recovery process can increase failover time. When the failover is complete, it can take additional time for the RDS console UI to reflect the new Availability Zone.

The failover mechanism automatically changes the DNS record of the DB instance to point to the standby DB instance. As a result, you will need to re-establish any existing connections to your DB instance. Due to how the Java DNS caching mechanism works, you may need to reconfigure your JVM environment. For more information on how to manage a Java application that caches DNS values in the case of a failover, see the AWS SDK for Java.

Amazon RDS handles failovers automatically so you can resume database operations as quickly as possible without administrative intervention. The primary DB instance switches over automatically to the standby replica if any of the following conditions occur:

- An Availability Zone outage
- The primary DB instance fails
- The DB instance's server type is changed
- The DB instance is undergoing software patching
- A manual failover of the DB instance was initiated using **Reboot with failover**

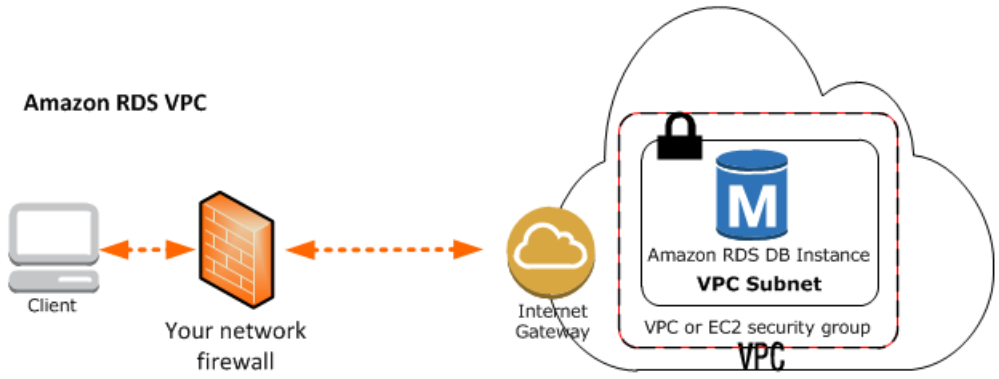
There are several ways to determine if your Multi-AZ DB instance has failed over:

- DB event subscriptions can be setup to notify you via email or SMS that a failover has been initiated. For more information about events, see Using Amazon RDS Event Notification (p. 575)
- You can view your DB events via the Amazon RDS console or APIs.
- You can view the current state of your Multi-AZ deployment via the Amazon RDS console and APIs.

For information on how you can respond to failovers, reduce recovery time, and other best practices for Amazon RDS, go to Best Practices for Amazon RDS (p. 54).

Amazon RDS and Amazon Virtual Private Cloud (VPC)

Amazon RDS lets you use the Amazon Virtual Private Cloud (VPC) service to create a virtual private cloud where you can launch a DB instance. When you use a virtual private cloud, you have control over your virtual networking environment: you can select your own IP address range, create subnets, and configure routing and access control lists. The basic functionality of Amazon RDS is the same whether it is running in a VPC or not: Amazon RDS manages backups, software patching, automatic failure detection, and recovery. There is no additional cost to run your DB instance in a VPC.



Amazon RDS supports two VPC platforms in each region: The *EC2-Classic* platform (shown as **EC2,VPC** in the RDS console) requires you to use the Amazon VPC service if you want to create a VPC, and the *EC2-VPC* platform (shown as **VPC** in the RDS console), which provides your AWS account with a default VPC in a region. If you are a new customer to Amazon RDS or if you are creating DB instances in a region you have not worked in before, chances are good you are on the *EC2-VPC* platform and that you have a default VPC. To determine which platform your account supports in a particular region, see Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 563).

For more information about using a VPC with Amazon RDS, see Using Amazon RDS with Amazon Virtual Private Cloud (VPC) (p. 562)

DB Instance Backups

Amazon RDS provides two different methods for backing up and restoring your Amazon DB instances: automated backups and DB snapshots. Automated backups automatically back up your DB instance during a specific, user-definable backup window, and keeps the backups for a limited, user-specified period of time (called the backup retention period); you can later recover your database to any point in time during that retention period. DB snapshots are user-initiated backups that enable you to back up your DB instance to a known state, and restore to that specific state at any time. Amazon RDS keeps all DB snapshots until you delete them.

Note
A brief I/O freeze, typically lasting a few seconds, occurs during both automated backups and DB snapshot operations on Single-AZ DB instances.

Automated Backup

Automated backup is an Amazon RDS feature that automatically creates a backup of your database. Automated backups are enabled by default for a new DB instance.

An automated backup occurs during a daily user-configurable period of time known as the preferred backup window. Backups created during the backup window are retained for a user-configurable number of days (the backup retention period). Note that if the backup requires more time than allotted to the backup window, the backup will continue to completion.

> **Note**
> An immediate outage will occur if you change the backup retention period from 0 to a non-zero value or from a non-zero value to 0.

The preferred backup window is the user-defined period of time during which your DB instance is backed up. Amazon RDS uses these periodic data backups in conjunction with your transaction logs to enable you to restore your DB instance to any second during your retention period, up to the *LatestRestorableTime* (typically up to the last five minutes). During the backup window, storage I/O may be suspended while your data is being backed up and you may experience elevated latency. This I/O suspension typically lasts for the duration of the snapshot. This period of I/O suspension is shorter for Multi-AZ DB deployments, since the backup is taken from the standby, but latency can occur during the backup process.

When the backup retention changes to a non-zero value, the first backup occurs immediately. Changing the backup retention period to 0 turns off automatic backups for the DB instance, and deletes all existing automated backups for the instance.

If you don't specify a preferred backup window when you create the DB instance, Amazon RDS assigns a default 30-minute backup window which is selected at random from an 8-hour block of time per region.

The following table lists the time blocks for each region from which the default backups windows are assigned.

Region	Time Block
US East (N. Virginia) region	03:00-11:00 UTC
US West (N. California) region	06:00-14:00 UTC
US West (Oregon) region	06:00-14:00 UTC
EU (Ireland) region	22:00-06:00 UTC
EU (Frankfurt) Region	23:00-07:00 UTC
Asia Pacific (Tokyo) Region	13:00-21:00 UTC
Asia Pacific (Sydney) Region	12:00-20:00 UTC
Asia Pacific (Singapore) Region	14:00-22:00 UTC
South America (São Paulo) Region	00:00-08:00 UTC
AWS GovCloud (US) Region	03:00-11:00 UTC

Changes to the backup window take effect immediately. The backup window cannot overlap with the weekly maintenance window for the DB instance.

When you delete a DB instance, you can create a final DB snapshot upon deletion; if you do, you can use this DB snapshot to restore the deleted DB instance at a later date. Amazon RDS retains this final user-created DB snapshot along with all other manually created DB snapshots after the DB instance is deleted. All automated backups are deleted and cannot be recovered when you delete a DB instance. Refer to the pricing page for information on backup storage costs.

For more information on working with automated backups, go to Working With Automated Backups (p. 496).

Point-In-Time Recovery

In addition to the daily automated backup, Amazon RDS archives database change logs. This enables you to recover your database to any point in time during the backup retention period, up to the last five minutes of database usage.

Amazon RDS stores multiple copies of your data, but for Single-AZ DB instances these copies are stored in a single availability zone. If for any reason a Single-AZ DB instance becomes unusable, you can use point-in-time recovery to launch a new DB instance with the latest restorable data. For more information on working with point-in-time recovery, go to Restoring a DB Instance to a Specified Time (p. 508).

Note
Multi-AZ deployments store copies of your data in different Availability Zones for greater levels of data durability. For more information on Multi-AZ deployments, see High Availability (Multi-AZ) (p. 71).

Automated Backups with Unsupported MySQL Storage Engines

Amazon RDS automated backups and DB snapshots are currently supported for all DB engines. For the MySQL DB engine, only the InnoDB storage engine is supported; use of these features with other MySQL storage engines, including MyISAM, may lead to unreliable behavior while restoring from backups. Specifically, since storage engines like MyISAM do not support reliable crash recovery, your tables can be corrupted in the event of a crash. For this reason, we encourage you to use the InnoDB storage engine.

If you choose to use MyISAM, you can attempt to manually repair tables that become damaged after a crash by using the REPAIR command ((see: http://dev.mysql.com/doc/refman/5.5/en/repair-table.html). However, as noted in the MySQL documentation, there is a good chance that you will not be able to recover all your data.

If you want to take DB snapshots with MyISAM tables, follow these steps:

Launch Process

1	Stop all activity to your MyISAM tables (that is, close all sessions)
2	Lock and flush each of your MyISAM tables
3	Issue a `CreateDBSnapshot` API call, or use the Amazon RDS CLI command `rds-create-db-snapshot`. When the snapshot has completed, release the locks and resume activity on the MyISAM tables. These steps force MyISAM to flush data stored in memory to disk thereby ensuring a clean start when you restore from a DB snapshot.

Finally, if you would like to convert existing MyISAM tables to InnoDB tables, you can use alter table command (for example, alter table *TABLE_NAME* engine=innodb;).

DB Snapshots

DB snapshots are user-initiated and enable you to back up your DB instance in a known state as frequently as you wish, and then restore to that specific state at any time. DB snapshots can be created with the Amazon RDS console or the `CreateDBSnapshot` action in the Amazon RDS API. DB snapshots are kept until you explicitly delete them with the Amazon RDS console or the `DeleteDBSnapshot` action in the Amazon RDS API. For more information on working with DB snapshots, see Creating a DB Snapshot (p. 499) and Restoring From a DB Snapshot (p. 501).

Related Topics

- Creating a DB Snapshot (p. 499)
- Restoring From a DB Snapshot (p. 501)
- Copying a DB Snapshot (p. 504)
- Working With Automated Backups (p. 496)

DB Instance Replication

Currently, you can create replicas of your DB instances in two ways. All DB instances can have a Multi-AZ deployment, where Amazon RDS automatically provisions and manages a standby replica in a different Availability Zone (independent infrastructure in a physically separate location). In the event of planned database maintenance, DB instance failure, or an Availability Zone failure, Amazon RDS will automatically failover to the standby so that database operations can resume quickly without administrative intervention. For more information on Multi-AZ deployments, see High Availability (Multi-AZ) (p. 71).

Amazon RDS also uses either the PostgreSQL or MySQL DB engine's built-in replication functionality to create a special type of DB instance called a Read Replica from a source DB instance. Updates made to the source DB instance are asynchronously copied to the Read Replica. You can reduce the load on your source DB instance by routing read queries from your applications to the Read Replica. Read replicas allow you to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. For more information about Read Replicas, see Working with PostgreSQL and MySQL Read Replicas (p. 472)

Storage for Amazon RDS

Amazon RDS uses Amazon Elastic Block Store (Amazon EBS) volumes for database and log storage. Depending on the amount of storage requested, Amazon RDS automatically stripes across multiple Amazon EBS volumes to enhance IOPS performance. Amazon RDS provides three types of storage with a range of storage and performance options.

Topics

- Amazon RDS Storage Types (p. 77)
- Performance Metrics (p. 78)
- Facts About Amazon RDS Storage (p. 78)
- General Purpose (SSD) Storage (p. 80)
- Amazon RDS Provisioned IOPS Storage to Improve Performance (p. 82)
- Factors That Affect Realized IOPS Rates (p. 85)

Amazon RDS Storage Types

Amazon RDS provides three storage types: magnetic, General Purpose (SSD), and Provisioned IOPS (input/output operations per second). They differ in performance characteristics and price, allowing you to tailor your storage performance and cost to the needs of your database. You can create MySQL, PostgreSQL, and Oracle RDS DB instances with up to 6TB of storage and SQL Server RDS DB instances with up to 4TB of storage when using the Provisioned IOPS and General Purpose (SSD) storage types. Existing MySQL, PostgreSQL, and Oracle RDS database instances can be scaled to these new database storage limits without any downtime. For a complete discussion of the different volume types, see the topic Amazon EBS Volume Types.

- **Magnetic (Standard)** – Magnetic storage, also called standard storage, offers cost-effective storage that is ideal for applications with light or burst I/O requirements. These volumes deliver approximately 100 IOPS on average, with burst capability of up to hundreds of IOPS, and they can range in size from 5 GB to 3 TB, depending on the DB instance engine that you chose. Magnetic storage is not reserved for a single DB instance, so performance can vary greatly depending on the demands placed on shared resources by other customers.
- **General Purpose (SSD)** – General purpose, SSD-backed storage, also called *gp2*, can provide faster access than disk-based storage. This storage type can deliver single-digit millisecond latencies, with a base performance of 3 IOPS/GB and the ability to burst to 3,000 IOPS for extended periods of time. In certain cases, based on your instance and storage configuration, you may get more than 3000 IOPS. General purpose (SSD) volumes can range in size from 5 GB to 6 TB for MySQL, PostgreSQL, and

Oracle DB instances, and from 5 GB to 4 TB for SQL Server DB instances. This storage type is excellent for small to medium-sized databases.

- **Provisioned IOPS** – Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that are sensitive to storage performance and consistency in random access I/O throughput. Provisioned IOPS volumes can range in size from 100 GB to 6 TB for MySQL, PostgreSQL, and Oracle DB engines. SQL Server Express and Web editions can range in size from 100 GB to 4 TB, while SQL Server Standard and Enterprise editions can range in size from 200 GB to 4 TB. You specify the amount of storage you want allocated, and then specify the amount of dedicated IOPS you want. These two values form a ratio, and this value maintains the ratio specified for the DB engine you chose. Amazon RDS delivers within 10 percent of the provisioned IOPS performance 99.9 percent of the time over a given year.

Several factors can affect the performance of Amazon EBS volumes, such as instance configuration, I/O characteristics, and workload demand. For more information about getting the most out of your Provisioned IOPS volumes, see Amazon EBS Volume Performance.

For existing MySQL, PostgreSQL, and Oracle DB instances, you might observe some I/O capacity improvement if you scale up your storage. Note that you cannot change the storage capacity of a SQL Server DB instance due to extensibility limitations of striped storage attached to a Windows Server environment.

Performance Metrics

Amazon RDS provides several metrics that you can use to determine how your DB instance is performing. You can view the metrics in the RDS console by selecting your DB instance and clicking Show Monitoring. You can also use Amazon CloudWatch to monitor these metrics. For more information, go to the Viewing DB Instance Metrics (p. 572).

- **IOPS** – the number of I/O operations completed per second. This metric is reported as the average IOPS for a given time interval. Amazon RDS reports read and write IOPS separately on one minute intervals. Total IOPS is the sum of the read and write IOPS. Typical values for IOPS range from zero to tens of thousands per second.
- **Latency** – the elapsed time between the submission of an I/O request and its completion. This metric is reported as the average latency for a given time interval. Amazon RDS reports read and write latency separately on one minute intervals in units of seconds. Typical values for latency are in the millisecond (ms); for example, Amazon RDS reports 2 ms as 0.002 seconds.
- **Throughput** – the number of bytes per second transferred to or from disk. This metric is reported as the average throughput for a given time interval. Amazon RDS reports read and write throughput separately on one minute intervals using units of megabytes per second (MB/s). Typical values for throughput range from zero to the I/O channel's maximum bandwidth.
- **Queue Depth** – the number of I/O requests in the queue waiting to be serviced. These are I/O requests that have been submitted by the application but have not been sent to the device because the device is busy servicing other I/O requests. Time spent waiting in the queue is a component of Latency and Service Time (not available as a metric). This metric is reported as the average queue depth for a given time interval. Amazon RDS reports queue depth in one minute intervals. Typical values for queue depth range from zero to several hundred.

Facts About Amazon RDS Storage

The following points are important facts you should know about Amazon RDS storage:

- The current maximum channel bandwidth available is 1000 megabits per second (Mbps) full duplex. In terms of the read and write throughput metrics, this equates to about 105 megabytes per second (MB/s) in each direction. A perfectly balanced workload of 50% reads and 50% writes may attain a maximum combined throughput of 210 MB/s. Note that this is channel throughput, which includes protocol overhead, so the actual data throughput may be less.
- Provisioned IOPS works with an I/O request size of 32 KB. An I/O request smaller than 32 KB is handled as one I/O; for example, 1000 16 KB I/O requests are treated the same as 1000 32 KB requests. I/O requests larger than 32 KB consume more than one I/O request; Provisioned IOPS consumption is a linear function of I/O request size above 32 KB. For example, a 48 KB I/O request consumes 1.5 I/O requests of storage capacity; a 64 KB I/O request consumes 2 I/O requests, etc. For more information about Provisioned IOPS, see Amazon RDS Provisioned IOPS Storage to Improve Performance (p. 82).

  Note that I/O size does not affect the IOPS values reported by the metrics, which are based solely on the number of I/Os over time. This means that it is possible to consume all of the IOPS provisioned with fewer I/Os than specified if the I/O sizes are larger than 32 KB. For example, a system provisioned for 5,000 IOPS can attain a maximum of 2,500 IOPS with 64 KB I/O or 1,250 IOPS with 128 KB IO.

  Note that magnetic storage does not provision I/O capacity, so all I/O sizes are counted as a single I/O. General purpose storage provisions I/O capacity based on the size of the volume. For more information on general purpose storage throughput, go to General Purpose (SSD) Volumes.
- The first time a DB instance is started and accesses an area of disk for the first time, the process can take longer than all subsequent accesses to the same disk area. This is known as the “first touch penalty.” Once an area of disk has incurred the first touch penalty, that area of disk **does not incur the penalty again for the life of the instance**, even if the DB instance is rebooted, restarted, or the DB instance class changes. Note that a DB instance created from a snapshot, a point-in-time restore, or a read replica is a new instance and does incur this first touch penalty.
- Because Amazon RDS manages your DB instance, we reserve overhead space on the instance. While the amount of reserved storage varies by DB instance class and other factors, this reserved space can be as much as one or two percent of the total storage.
- Provisioned IOPS provides a way to reserve I/O capacity by specifying IOPS. Like any other system capacity attribute, maximum throughput under load will be constrained by the resource that is consumed first. That resource could be IOPS, channel bandwidth, CPU, memory, or database internal resources.

Other Factors That Impact Storage Performance

All of the following system related activities consume I/O capacity and may reduce database instance performance while in progress:

- DB snapshot creation
- Nightly backups
- Multi-AZ peer creation
- Read replica creation
- Scaling storage

System resources can constrain the throughput of a DB instance, but there can be other reasons for a bottleneck. If you find the following situation, your database could be the issue:

- The channel throughput limit is not reached
- Queue depths are consistently low
- CPU utilization is under 80%
- There is free memory available
- There is no swap activity
- There is plenty of free disk space

- Your application has dozens of threads all submitting transactions as fast as the database will take them, but there is clearly unused I/O capacity

If there isn't at least one system resource that is at or near a limit, and adding threads doesn't increase the database transaction rate, the bottleneck is most likely contention in the database. The most common forms are row lock and index page lock contention, but there are many other possibilities. If this is your situation, you should seek the advice of a database performance tuning expert.

Adding Storage and Changing Storage Type

You can modify a DB instance to use additional storage and you can convert to a different storage type. Adding storage or converting to a different storage type can take time and reduces the performance of your DB instance, so you should plan when to make these changes.

Although your DB instance is available for reads and writes when adding storage, you may experience degraded performance until the process is complete. Adding storage may take several hours; the duration of the process depends on several factors such as database load, storage size, storage type, amount of IOPS provisioned (if any), and number of prior scale storage operations. Typical scale storage times will be under 24 hours, but can take up to several days in some cases. During the scaling process, the DB instance will be available for use, but may experience performance degradation.

Storage conversions between magnetic storage and general purpose (SSD) storage can potentially deplete the initial 5.4 million I/O credits (3,000 IOPS X 30 Minutes) allocated for general purpose (SSD) storage. When performing these storage conversions, the first 82 GB of data will be converted at approximately 3,000 IOPS, while the remaining data will be converted at the base performance rate of 3 IOPS per GB of allocated general purpose (SSD) storage. This can result in longer conversion times. You can provision more general purpose (SSD) storage to increase your base I/O performance rate, thus improving the conversion time, but note that you cannot reduce storage size once it has been allocated.

General Purpose (SSD) Storage

General purpose (SSD) storage offers cost-effective storage that is ideal for small or medium-sized database workloads. This storage type can deliver single-digit millisecond latencies, with a base performance of 3 IOPS/GB and the ability to burst to 3,000 IOPS for extended periods of time. In certain cases, based on your instance and storage configuration, you may get more than 3000 IOPS. General purpose (SSD) storage volumes can range in size from 5 GB to 6 TB for MySQL, PostgreSQL, and Oracle DB instances and from 5 GB to 4 TB for SQL Server DB instances. Note that provisioning less than 100 GB of general purpose (SSD) storage for high-throughput workloads can result in higher latencies if the initial general purpose (SSD) I/O credit balance is depleted.

I/O Credits and Burst Performance

General Purpose (SSD) storage performance is governed by volume size, which dictates the base performance level of the volume and how quickly it accumulates I/O credits. Larger volumes have higher base performance levels and accumulate I/O credits faster. *I/O credits* represent the available bandwidth that your General Purpose (SSD) storage can use to burst large amounts of I/O when more than the base level of performance is needed. The more credits your storage has for I/O, the more time it can burst beyond its base performance level and the better it performs when more performance is needed.

When using General Purpose (SSD) storage, your DB instance receives an initial I/O credit balance of 5.4 million I/O credits, which is enough to sustain the maximum burst performance of 3,000 IOPS for 30 minutes. This initial credit balance is designed to provide a fast initial boot cycle for boot volumes and to provide a good bootstrapping experience for other applications. Your storage earns I/O credits every

second at a base performance rate of 3 IOPS per GB of volume size. For example, a 100 GB General Purpose (SSD) storage has a base performance of 300 IOPS.

When your storage requires more than the base performance I/O level, it uses I/O credits in the credit balance to burst to the required performance level, up to a maximum of 3,000 IOPS. Storage larger than 1,000 GB has a base performance that is equal or greater than the maximum burst performance, so its I/O credit balance never depletes and it can burst indefinitely. When your storage uses fewer I/O credits than it earns in a second, unused I/O credits are added to the I/O credit balance. The maximum I/O credit balance for a DB instance using General Purpose (SSD) storage is equal to the initial credit balance (5.4 million I/O credits).

If your storage uses all of its I/O credit balance, its maximum performance will remain at the base performance level (the rate at which your storage earns credits) until I/O demand drops below the base level and unused credits are added to the I/O credit balance. The more storage, the greater the base performance is and the faster it replenishes the credit balance.

Note

Storage conversions between Magnetic storage and General Purpose (SSD) storage can potentially deplete the initial 5.4 million I/O credits (3,000 IOPS X 30 Minutes) allocated for General Purpose (SSD) storage. When performing these storage conversions, the first 82 GB of data will be converted at approx. 3,000 IOPS, while the remaining data will be converted at the base performance rate of 3 IOPS per GB of allocated General Purpose (SSD) storage. This can result in longer conversion times. You can provision more General Purpose (SSD) storage to increase your base I/O performance rate, thus improving the conversion time, but note that you cannot reduce storage size once it has been allocated.

The following table lists several storage sizes and the associated base performance of the storage (which is also the rate at which it accumulates I/O credits), the burst duration at the 3,000 IOPS maximum (when starting with a full credit balance), and the time in seconds that the storage takes to refill an empty credit balance.

Storage size (GB)	Base performance (IOPS)	Maximum burst duration @ 3,000 IOPS (seconds)	Seconds to fill empty credit balance
1	3	1,802	1,800,000
100	300	2,000	18,000
250	750	2,400	7,200
500	1,500	3,600	3,600
750	2,250	7,200	2,400
1,000	3,000	Infinite	N/A

The burst duration of your storage depends on the size of the storage, the burst IOPS required, and the credit balance when the burst begins. This relationship is shown in the equation below:

$$\text{Burst duration} = \frac{\text{(Credit balance)}}{\text{(Burst IOPS) - 3(Storage size in GB)}}$$

If you notice that your storage performance is frequently limited to the base level due to an empty I/O credit balance, you should consider allocating more General Purpose (SSD) storage with a higher base

performance level. Alternatively, you can switch to Provisioned IOPS storage for workloads that require sustained IOPS performance greater than 3,000 IOPS.

For workloads with steady state I/O requirements, provisioning less than 100 GB of General Purpose (SSD) storage may result in higher latencies if you exhaust your I/O burst credit balance.

Amazon RDS Provisioned IOPS Storage to Improve Performance

For any production application that requires fast and consistent I/O performance, we recommend Provisioned IOPS (input/output operations per second) storage. Provisioned IOPS storage is a storage type that delivers fast, predictable, and consistent throughput performance. When you create a DB instance, you specify an IOPS rate and storage space allocation. Amazon RDS provisions that IOPS rate and storage for the lifetime of the DB instance or until you change it. Provisioned IOPS storage is optimized for I/O intensive, online transaction processing (OLTP) workloads that have consistent performance requirements. Provisioned IOPS helps performance tuning.

> **Note**
> You cannot decrease storage allocated for a DB instance.

Topics

- Using Provisioned IOPS Storage with Multi-AZ, Read Replicas, Snapshots, VPC, and DB Instance Classes (p. 83)
- Provisioned IOPS Storage Costs (p. 83)
- Getting the Most out of Amazon RDS Provisioned IOPS (p. 84)
- Provisioned IOPS Storage Support in the CLI and Amazon RDS API (p. 84)

You can create a DB instance that uses Provisioned IOPS storage by using the AWS Management Console, the Amazon RDS API, or the Command Line Interface (CLI). You specify the IOPS rate and the amount of storage that you require. You can provision a MySQL, PostgreSQL, or Oracle DB instance with up to 30,000 IOPS and 6 TB of allocated storage. You can provision a SQL Server DB instance with up to 20,000 IOPS and 4 TB of allocated storage.

> **Note**
> Your actual realized IOPS may vary from the value that you specify depending on your database workload, DB instance size, and the page size and channel bandwidth that are available for your DB engine. For more information, see Factors That Affect Realized IOPS Rates (p. 85).

The ratio of the requested IOPS rate to the amount of storage allocated is important. The ratio of IOPS to storage, in GB, for your DB instances should be between 3:1 and 10:1 for MySQL, PostgreSQL, SQL Server (excluding SQL Server Express), and Oracle DB instances. For example, you could start by provisioning an Oracle DB instance with 1000 IOPS and 200 GB storage (a ratio of 5:1). You could then scale up to 2000 IOPS with 200 GB of storage (a ratio of 10:1), 3000 IOPS with 300 GB of storage, and up to the maximum for an Oracle DB instance of 30,000 IOPS with 6 TB (6000 GB) of storage (a ratio of 5:1).

The following table shows the IOPS and storage range for each database engine.

	Range of Provisioned IOPS	Range of Storage	Range of IOPS to Storage (GB) Ratio
MySQL	1000 - 30,000 IOPS	100 GB - 6 TB	3:1 - 10:1

	Range of Provisioned IOPS	Range of Storage	Range of IOPS to Storage (GB) Ratio
PostgreSQL	1000 - 30,000 IOPS	100 GB - 6 TB	3:1 - 10:1
Oracle	1000 - 30,000 IOPS	100 GB - 6 TB	3:1 - 10:1
SQL Server Express and Web	1000 - 20,000 IOPS	100 GB - 4 TB	3:1 - 10:1
SQL Server Standard and Enterprise	1000 - 20,000 IOPS	200 GB - 4 TB	3:1 - 10:1

You can modify an existing Oracle or MySQL DB instance to use Provisioned IOPS storage, and you can modify Provisioned IOPS storage settings.

Using Provisioned IOPS Storage with Multi-AZ, Read Replicas, Snapshots, VPC, and DB Instance Classes

For production OLTP use cases, we recommend that you use Multi-AZ deployments for enhanced fault tolerance and Provisioned IOPS storage for fast and predictable performance. In addition to Multi-AZ deployments, Provisioned IOPS storage complements the following features:

- Amazon VPC for network isolation and enhanced security.
- Read Replicas – The type of storage on a read replica is independent of that on the master DB instance. For example, if the master DB instance uses magnetic storage, you can add read replicas that use Provisioned IOPS storage and vice versa. If you use magnetic storage–based read replicas with a master DB instance that uses Provisioned IOPS storage, the performance of your read replicas may differ considerably from that of a configuration in which both the master DB instance and the read replicas are using Provisioned IOPS storage.
- DB Snapshots – If you are using a DB instance that uses Provisioned IOPS storage, you can use a DB snapshot to restore an identically configured DB instance, regardless of whether the target DB instance uses magnetic storage or Provisioned IOPS storage. If your DB instance uses magnetic storage, you can use a DB snapshot to restore only a DB instance that uses magnetic storage.
- You can use Provisioned IOPS storage with any DB instance class. However, smaller DB instance classes will not consistently make the best use of Provisioned IOPS storage. For the best performance, we recommend that you use one of the DB instance types that are optimized for Provisioned IOPS storage.

Provisioned IOPS Storage Costs

Because Provisioned IOPS storage reserves resources for your use, you are charged for the resources whether or not you use them in a given month. When you use Provisioned IOPS storage, you are not charged the monthly Amazon RDS I/O charge. If you prefer to pay only for I/O that you consume, a DB instance that uses magnetic storage may be a better choice. For Amazon RDS pricing information, see the Amazon RDS product page.

Getting the Most out of Amazon RDS Provisioned IOPS

Using Provisioned IOPS storage increases the number of I/O requests the system is capable of processing concurrently. Increased concurrency allows for decreased latency since I/O requests spend less time in a queue. Decreased latency allows for faster database commits, which improves response time and allows for higher database throughput.

For example, consider a heavily loaded OLTP database provisioned for 10,000 Provisioned IOPS that runs consistently at the channel limit of 105 Mbps throughput for reads. The workload isn't perfectly balanced, so there is some unused write channel bandwidth. The instance would consume less than 10,000 IOPS and but would still benefit from increasing capacity to 20,000 Provisioned IOPS.

Increasing Provisioned IOPS capacity from 10,000 to 20,000 doubles the system's capacity for concurrent I/O. Increased concurrency means decreased latency, which allows transactions to complete faster, so the database transaction rate increases. Read and write latency would improve by different amounts and the system would settle into a new equilibrium based on whichever resource becomes constrained first.

It is possible for Provisioned IOPS consumption to actually *decrease* under these conditions even though the database transaction rate can be much higher. For example, you could see write requests decline accompanied by an increase in write throughput. That's a good indicator that your database is making better use of group commit. More write throughput and the same write IOPS means log writes have become larger but are still less than 256 KB. More write throughput and fewer write I/O means log writes have become larger and are averaging larger than 32 KB since those I/O requests consume more than one I/O of Provisioned IOPS capacity.

Provisioned IOPS Storage Support in the CLI and Amazon RDS API

The Amazon RDS CLI supports Provisioned IOPS storage in the following commands:

- **rds-create-db-snapshot** – The output shows the IOPS value.
- **rds-create-db-instance** – Includes the input parameter *`iops`*, and the output shows the IOPS rate.
- **rds-modify-db-instance** – Includes the input parameter *`iops`*, and the output shows the IOPS rate.
- **rds-restore-db-instance-from-db-snapshot** – Includes the input parameter *`iops`*, and the output shows current IOPS rate. If **Apply Immediately** was specified, the output also shows the pending IOPS rate.
- **rds-restore-db-instance-to-point-in-time** – Includes the input parameter *`iops`*, and the output shows the IOPS rate.
- **rds-create-db-instance-read-replica** – Includes the input parameter *`iops`*, and the output shows the IOPS rate.

The Amazon RDS API supports Provisioned IOPS storage in the following actions:

- `CreateDBInstance` – Includes the input parameter *`iops`*, and the output shows the IOPS rate.
- `CreateDBInstanceReadReplica` – Includes the input parameter *`iops`*, and the output shows the IOPS rate.
- `CreateDBSnapshot` – The output shows the IOPS rate.
- `ModifyDBInstance` – Includes the input parameter *`iops`*, and the output shows the IOPS rate.
- `RestoreDBInstanceFromDBSnapshot` – Includes the input parameter *`iops`*, and the output shows current IOPS rate. If **Apply Immediately** was specified, the output also shows the pending IOPS rate.

- `RestoreDBInstanceToPointInTime` – Includes the input parameter *`iops`*, and the output shows the IOPS rate.

Factors That Affect Realized IOPS Rates

Your actual realized IOPS rate may vary from the amount that you provision depending on page size and network bandwidth, which are determined in part by your DB engine. It is also affected by DB instance size and database workload.

Page Size and Channel Bandwidth

The theoretical maximum IOPS rate is also a function of database I/O page size and available channel bandwidth. MySQL uses a page size of 16 KB, while Oracle, PostgreSQL (default), and SQL Server use 8 KB. On a DB instance with a full duplex I/O channel bandwidth of 1000 megabits per second (Mbps), the maximum IOPS for page I/O is about 8,000 IOPS total for both directions (input/output channel) for 16 KB I/O and 16,000 IOPS total for both directions for 8 KB I/O.

If traffic on one of the channels reaches capacity, available IOPS on the other channel cannot be reallocated. As a result, the attainable IOPS rate will be less than the provisioned IOPS rate.

Each page read or write constitutes one I/O operation. Database operations that read or write more than a single page will use multiple I/O operations for each database operation. I/O requests larger than 32 KB are treated as more than one I/O for the purposes of PIOPS capacity consumption. A 40 KB I/O request will consume 1.25 I/Os, a 48 KB request will consume 1.5 I/Os, a 64 KB request will consume 2 I/Os, and so on. The I/O request is not split into separate I/Os; all I/O requests are presented to the storage device unchanged. For example, if the database submits a 128 KB I/O request, it goes to the storage device as a single 128 KB I/O request, but it will consume the same amount of PIOPS capacity as four 32 KB I/O requests.

The following table shows the page size and the theoretical maximum IOPS rate for each DB engine. IOPS rates are based on the m2.4xlarge instance class (for Oracle and SQL Server) or the cr1.8xlarge instance class (for MySQL and PostgreSQL) with full duplex and a workload that is perfectly balanced between reads and writes.

DB Engine	Page Size	Theoretical Maximum IOPS Rate
MySQL	16 KB	30,000
PostgreSQL	8 KB	30,000
Oracle	8 KB	25,000
SQL Server	8 KB	20,000

Note
If you provision an IOPS rate that is higher than the maximum or that is higher than your realized IOPS rate, you may still benefit from reduced latency and improvements in overall throughput.

DB Instance Class

If you are using Provisioned IOPS storage, we recommend that you use the db.m3.xlarge, db.m3.2xlarge, db.m1.large, db.m1.xlarge, db.m2.2xlarge, db.m2.4xlarge, db.r3.xlarge, db.r3.2xlarge, or db.r3.4xlarge DB instance classes. These instance classes are optimized for Provisioned IOPS storage; other instance classes are not. You can also effectively use the high-memory-cluster instance classes: db.r3.8xlarge

and db.cr1.8xlarge for high-performance applications, though these two classes are not optimized for Provisioned IOPS.

DB Instance Classes Optimized for Provisioned IOPS	Dedicated EBS Throughput (Mbps)	Maximum 16k IOPS Rate**	Max Bandwidth (MB/s)**
db.m1.large	500 Mbps	4000	62.5
db.m1.xlarge	1000 Mbps	8000	125
db.m2.2xlarge	500 Mbps	4000	62.5
db.m2.4xlarge	1000 Mbps	8000	125
db.m3.xlarge	500 Mbps	4000	62.5
db.m3.2xlarge	1000 Mbps	8000	125
db.r3.xlarge	500 Mbps	4000	62.5
db.r3.2xlarge	1000 Mbps	8000	125
db.r3.4xlarge	2000 Mbps	16000	250

** This value is a rounded approximation based on a 100% read-only workload and it is provided as a baseline configuration aid. EBS-optimized connections are full-duplex, and can drive more throughput and IOPS in a 50/50 read/write workload where both communication lanes are used. In some cases, network and file system overhead can reduce the maximum throughput and IOPS available.

Database Workload

System activities such as automated backups, DB snapshots, and scale storage operations may consume some I/O, which will reduce the overall capacity available for normal database operations. If your database design results in concurrency issues, locking, or other forms of database contention, you may not be able to directly use all the bandwidth that you provision.

If you provision IOPS capacity to meet your peak workload demand, during the non-peak periods, your application will probably consume fewer IOPS on average than provisioned.

To help you verify that you are making the best use of your Provisioned IOPS storage, we have added a new CloudWatch Metric called Disk Queue Depth. If your application is maintaining an average queue depth of approximately 5 outstanding I/O operations per 1000 IOPS that you provision, you can assume that you are consuming the capacity that you provisioned. For example, if you provisioned 10,000 IOPS, you should have a minimum of 50 outstanding I/O operations in order to use the capacity you provisioned.

Security in Amazon RDS

Topics

- Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources (p. 88)
- Encrypting Amazon RDS Resources (p. 106)
- Using SSL to Encrypt a Connection to a DB Instance (p. 109)
- Amazon RDS Security Groups (p. 110)

You can manage access to your Amazon Relational Database Service(Amazon RDS) resources and your databases on a DB instance. The method you use to manage access depends on what type of task the user needs to perform with Amazon RDS:

- Run your DB instance in an Amazon Virtual Private Cloud (VPC) for the greatest possible network access control. For more information about creating a DB instance in a VPC, see Using Amazon RDS with Amazon Virtual Private Cloud (VPC).
- Use AWS Identity and Access Management (IAM) policies to assign permissions that determine who is allowed to manage RDS resources. For example, you can use IAM to determine who is allowed to create, describe, modify, and delete DB instances, tag resources, or modify DB security groups. For information on setting up a IAM user, see Create an IAM User (p. 8)
- Use security groups to control what IP addresses or EC2 instances can connect to your databases on a DB instance. When you first create a DB instance, its firewall prevents any database access except through rules specified by an associated security group.
- Use Secure Socket Layer (SSL) connections with DB instances running the MySQL, PostgreSQL, or Microsoft SQL Server database engines; for more information on using SSL with a DB instance, see Using SSL to Encrypt a Connection to a DB Instance (p. 109).
- Use RDS encryption to secure your RDS instances and snapshots at rest. RDS encryption uses the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your RDS instance. For more information, see Encrypting Amazon RDS Resources (p. 106).
- Use network encryption and transparent data encryption with Oracle DB instances; for more information, see Oracle Native Network Encryption (p. 233) and Oracle Transparent Data Encryption (TDE) (p. 235)
- Use the security features of your DB engine to control who can log in to the databases on a DB instance, just as you would if the database was on your local network.

Note
You only have to configure security for your use cases; you do not have to configure security access for processes that Amazon RDS manages, such as creating backups, replicating data between a master and a Read Replica, or other processes.

Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources

You can use AWS IAM to create permissions that specify which Amazon RDS actions a user, group, or role in your AWS account can perform, and on which RDS resources those actions can be performed. You specify permissions using an IAM policy, which is a JSON document.

When you sign up for an AWS account, you receive account access that lets you use AWS IAM to create users and grant them specific permissions to Amazon RDS actions and resources your users have access to. Your account access lets you create a DB instance and provide a master user name and master password to the instance. You use the master user name and master password to access the DB instance and create database resources and to set up users on the DB instance.

You should not share your AWS account information with anyone. For more information about managing your AWS account information, see Best Practices for Managing AWS Keys and IAM Best Practices.

You can create permissions that manage access to the following Amazon RDS resources:

- DB instances
- DB snapshots
- Read replicas
- Reserved instances
- DB security groups
- DB option groups
- DB parameter groups
- Event subscriptions
- DB subnet groups

To manage access to your Amazon RDS resources, you should take the following steps:

1. Create IAM users (user identities) under your AWS account for all users who will manage your Amazon RDS resources. Each user can have a separate password (for access using the Amazon RDS Management Console) and access keys (for programmatic and command line interface access). You can organize IAM users into groups, which makes it easier to manage permissions for multiple users at a time.
2. Determine what tasks each user and group will have regarding your Amazon RDS resources. For example, you could have groups for administrators, security personnel, DBAs, and developers.
3. Optionally, you can tag the Amazon RDS resources you want to control access to. You can assign a tag, a key-value pair, to any Amazon RDS resource, and use that tag as a way to specify a particular resource in an IAM policy.
4. Create the IAM policies that define the actions a user can take, and specify the Amazon RDS resources required for each task using Amazon Resource Names (ARNs). If you have used tags for your Amazon RDS resources, you can add conditions to the policy to test for those tag values.
5. Attach the policies to the applicable users or groups.

Creating IAM Policies for Amazon RDS

By default, newly created IAM users do not have permission to access any AWS resources. This means that IAM users also can't use the Amazon RDS console or CLI. To allow IAM users to use the features of Amazon RDS, you must create IAM policies that allow users to access the required Amazon RDS API actions and resources, and then attach the policies to the IAM users or groups that require those permissions.

An IAM policy is a JSON document that consists of one or more statements. Each statement in an IAM policy is made up of elements that define what actions can be taken on what resources. The following example shows a simple policy statement that allows a user to only create a DB instance that must have "test" prefixed to the DB instance name, use the MySQL DB engine, and can only use the micro DB instance class.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds:CreateDBInstance",
            "Resource": "arn:aws:rds:us-east-1:1234567890:db:test*",
            "Condition": {
                "StringEquals": {
                    "rds:DatabaseEngine": "mysql",
                    "rds:DatabaseClass": "db.t1.micro"
                }
            }
        }
    ]
}
```

The `Version` element is required, and the value must be "2012-10-17". The `Effect` element is set to either "Allow" or "Deny" . (Actions are denied by default, so you typically specify "Allow".) The `Action` element lists which AWS APIs the policy will allow (or deny). In this case, the `Action` element lists one action from the Amazon RDS API, so it will be the only action allowed by this policy statement. Note that the action is identified by both service name (rds) and action (CreateDBInstance); policies can list actions from any AWS service. You can use wildcards (*) to specify actions—for example, the action rds:Describe* would allow the user to perform any Amazon RDS action that begins with Describe (DescribeDBInstances, DescribeDBLogFiles, DescribeDBParameterGroups, DescribeDBSnapshots, etc.) .

> **Note**
> The rds-download-db-logfile CLI command calls both the DownloadDBLogFilePortion API and the DownloadCompleteDBLogFile (p. 637) REST API, depending on the parameters that you specify. If you need to create an IAM policy that applies to both API calls, then specify an action that applies to both by using `rds:Download*`.

The `Resource` element lets you specify which resources the user can perform the actions on or with. In this example, the user can only create DB instances that have the prefix "test" in the DB instance name. You specify resources using an Amazon Resources Name (ARN) that includes the name of the service that the resource belongs to (rds), the region (us-east-1), the account number, and the type of resource (a DB instance). For more information on creating ARNs, see Constructing an Amazon RDS Amazon Resource Name (ARN) (p. 493)

Finally, the optional `Condition` element lets you specify additional restrictions on the policy, such as date/time, source IP address, region, or tags. In this example, the `Condition` element indicates that the

actions are allowed only on instances with the MySQL DB engine and the micro DB instance class. For more information on creating conditions, see Condition.

This policy might be attached to an individual IAM user, and in that case, that user would be allowed to perform the listed actions. You could instead attach the policy to an IAM group, and then every IAM user in that group would have these permissions. You can also attach the policy to a role so that delegated or federated users could perform the action.

Permissions Needed to Use the Amazon RDS Console

When users work with the Amazon RDS console, you must grant them permissions not only to perform the specific actions that you want to allow, but also permissions to actions that the console itself needs. For example, simply to list resources, the console runs the API actions such as `DescribeSecurityGroups` and `DescribeSubnets`. Users working in the console must have these permissions; if they don't, portions of the console that users need to work with might simply display a message that users don't have permissions for a task.

The following example policy statement shows permissions that users typically need in order to work in the Amazon RDS console. Notice that this includes RDS actions that start with the word "Describe," a number of EC2 and CloudWatch actions that likewise pertain to describing (listing) resources, and all SNS actions. The policy allows these actions for all resources owned by the account.

```
{
    "Version":"2012-10-17",
    "Statement":[{
    "Effect": "Allow",
    "Action": [
        "rds:Describe*",
        "rds:ListTagsForResource",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeVpcs",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:DescribeAlarms",
        "sns:*"
            ],
   "Resource": "*"
   }]
}
```

How Resource Authorization Works in Amazon RDS

When a user requests an Amazon RDS action, an IAM authorization request is generated for every resource identified in the request. Amazon RDS checks the IAM policy for the user who is making the request. If the policy explicitly allows the user to perform the requested action on the specified resources, then the action is allowed; otherwise, the action is denied.

An authorization request that applies to multiple resources can result in multiple resource authorizations. For example, a point-in-time-restore to a new database instance will generate two authorization requests:

1. An authorization request will be generated for the target database instance.
2. An authorization request will be generated for the snapshot that is being restored.

Note that a policy can be used to limit the possible values a resource can have. For example, storage or compute size can be limited to specific values or ranges. For a fuller explanation about how an IAM policy is evaluated, see IAM Policy Evaluation Logic.

Specifying Conditions in an IAM Policy for Amazon RDS

When creating an IAM policy, you can specify conditions in two ways. You can create a condition that is based on a tag associated with a resource, or you can use a predefined key, such as the DB engine type or the DB engine class. The following tables shows the predefined keys you can use when defining IAM policy for Amazon RDS. Note that tag key/value pairs and predefined keys are case sensitive.

For more information and a list of which RDS APIs support resource-level permissions, see Supported Resource-Level Permissions for Amazon RDS API Actions (p. 96).

AWS Predefined Keys

AWS provides several predefined keys that apply to all AWS resources that support IAM policies. The following table shows the AWS predefined keys that apply to Amazon RDS resources.

AWS Predefined Key	Description	Value type
aws:CurrentTime	The current time. Used for date conditions.	Date/Time
aws:EpochTime	The current time in epoch or UNIX time format. Used for date conditions.	Date/Time
aws:principaltype	The type of principal (user, account, federated user, etc.) for the current request.	String
aws:SourceIp	The requester's IP address (see IP Address). Note that if you use aws:SourceIp, and the request comes from an Amazon EC2 instance, the instance's public IP address is evaluated.	IP Address
aws:UserAgent	The requester's client application.	String
aws:userid	The requester's user ID.	String
aws:username	The requester's user name	String

Amazon RDS Predefined Keys

Amazon RDS also has predefined keys that you can include in Condition elements in an IAM policy. The following table shows the Amazon RDS predefined keys that apply to Amazon RDS resources.

RDS Predefined Key	Description	Value type
rds:DatabaseClass	The DB instance class of a DB instance	String
rds:DatabaseEngine	The DB engine of the DB instance	String
rds:DatabaseName	The name of the database on the DB instance	String

RDS Predefined Key	Description	Value type
rds:MultiAz	Specify whether the DB instance runs in multiple availability zones. 1 indicates that the DB instance is using multi-AZ.	Integer
rds:Piops	This key will be present when a request is made for a DB instance with PIOPS enabled. The value will contain the number of provisioned IOPS that an instance supports. 0 indicates a DB instance that does not have PIOPS enabled.	Integer
rds:StorageSize	The storage volume size (in GB)	Integer
rds:Vpc	Specify whether the DB instance runs in a virtual private cloud	Boolean

For example, the following `Condition` element uses a predefined key and specifies that the condition applies to the DB engine MySQL:

```
"Condition":{"StringEquals":{"rds:DatabaseEngine": "mysql" } }
```

Using Custom Tags with a Condition Element

You can also create policies using your own custom tag names and values. For example, if you added a tag named `environment` to your DB instances with values such as "beta", "staging", "production", and so on, you could create a policy that restricts certain users to DB instances based on the `environment` tag value. The syntax for a custom tag condition is as follows:

```
"Condition":{"StringEquals":{"rds:rds-tag-identifier/tag-name": ["value"]} }
```

Important
If you are managing access to your RDS resources using tagging, then we recommend that you secure access to the tags for your RDS resources. You can manage access to tags by creating policies for the `AddTagsToResource` and `RemoveTagsFromResource` actions. For example, the following policy denies users the ability to add or remove tags for all resources. You can then create policies to allow specific users to add or remove tags.

```
{
    "Version":"2012-10-17",
    "Statement":[{
      "Effect":"Deny",
      "Action": ["rds:AddTagsToResource","rds:RemoveTagsFromResource"],
      "Resource": "*"
    }]
}
```

For information on creating tags, see Tagging Amazon RDS Resources (p. 486).

You can use the following RDS tag identifiers in a `Condition` element.

RDS tag identifier	Applies to
db-tag	DB instances, including Read Replicas.
snapshot-tag	DB snapshots.
ri-tag	Reserved DB instances.

RDS tag identifier	Applies to
secgrp-tag	DB security groups.
og-tag	DB option groups.
pg-tag	DB parameter groups.
subgrp-tag	DB subnet groups.
es-tag	Event subscriptions.

For example, the following `Condition` element applies to DB instances with a tag named **`environment`** and a tag value of **`production`**.

```
"Condition":{"StringEquals":{"rds:db-tag/environment": ["production"]} }
```

For more information about the IAM policy `Condition` element, see Condition.

Example IAM Policies for Amazon RDS

The following examples show simple IAM policy statements that you can use to manage the access IAM users have to Amazon RDS resources.

Topics

- Example 1: Permit a user to perform any Describe action on any RDS resource (p. 93)
- Example 2: Permit a user to create a DB instance that uses a specified DB engine (p. 94)
- Example 3: Permit a user to create a DB instance that uses the specified DB parameter and security groups (p. 94)
- Example 4: Prevent a user from creating a DB instance that uses specified DB parameter groups (p. 94)
- Example 5: Prevent users from creating DB instances for certain DB instance classes and from creating DB instances that use Provisioned IOPS. (p. 95)
- Example 6: Permits a user to perform an action on a resource tagged with two different values (p. 95)
- Example 7:Permits a user to perform actions on a DB instance with a DB instance name prefixed with the user name (p. 96)
- Example 8: Prevent a user from deleting a DB instance (p. 96)

Example 1: Permit a user to perform any Describe action on any RDS resource

The following statement allows a user to run all the actions whose names begin with "Describe," which shows information about an RDS resource such as a DB instance. Note that the “*” in the `Resource` element indicates that the actions are allowed for all Amazon RDS resources owned by the account.

```
{
"Version":"2012-10-17",
"Statement":[{
"Effect":"Allow",
"Action":"rds:Describe*",
"Resource":"*"
  }]
}
```

Example 2: Permit a user to create a DB instance that uses a specified DB engine

The following statement uses a predefined Amazon RDS key and allows a user to create only DB instances that use the MySQL DB engine. The `Condition` element indicates that the DB engine requirement is MySQL.

```
{
"Version":"2012-10-17",
"Statement":[{
"Effect":"Allow",
"Action": "rds:CreateDBInstance",
"Resource": "*",
"Condition":{"StringEquals":{"rds:DatabaseEngine":"mysql"}}
  }]
}
```

Example 3: Permit a user to create a DB instance that uses the specified DB parameter and security groups

The following statement allows a user to only create a DB instance that must use the *mysql-production* DB parameter group and the *db-production* DB security group.

```
{
"Version":"2012-10-17",
"Statement":[{
"Effect":"Allow",
"Action": "rds:CreateDBInstance",
"Resource": [
    "arn:aws:rds:us-east-1:1234567890:pg:mysql-production",
    "arn:aws:rds:us-east-1:1234567890:secgrp:db-production" ]
  }]
}
```

Example 4: Prevent a user from creating a DB instance that uses specified DB parameter groups

The following statement prevents a user from creating a DB instance that uses DB parameter groups with specific tag values. You might apply this policy if you require that a specific customer-created DB parameter group always be used when creating DB instances. Note that statements that use Deny are most often used to restrict access that was granted by a broader statement.

```
{
"Version":"2012-10-17",
"Statement":[{
"Effect":"Deny",
"Action": "rds:CreateDBInstance",
"Resource": "*",
"Condition": {"StringEquals": {"rds:pg-tag/usage" : "prod" } }
```

```
  }]
}
```

Example 5: Prevent users from creating DB instances for certain DB instance classes and from creating DB instances that use Provisioned IOPS.

The following statement prevents users from creating DB instances that use the DB instance classes m2.2xlarge and m2.4xlarge, which are the largest and most expensive instances. This example also prevents users from creating DB instances that use Provisioned IOPS, which is an additional cost.

```
 {
"Version":"2012-10-17",
"Statement":[
{
"Effect":"Deny",
"Action":"rds:CreateDBInstance",
"Resource": "*",
"Condition":{"StringEquals":{"rds:DatabaseClass":["db.m2.4xlarge",
"db.m2.2xlarge"]}}
},
{
"Effect":"Deny",
"Action":"rds:CreateDBInstance",
"Resource": "*",
"Condition":{"NumericNotEquals":{"rds:Piops":"0"}}
}
]
}
```

You can add a tag to an Amazon RDS resource, and then use that tag in a policy to specify a particular resource. The following examples use Amazon RDS resource tags as part of the IAM policy to specify a particular resource.

For more information about adding tags to an Amazon RDS resource, see Constructing an Amazon RDS Amazon Resource Name (ARN) (p. 493). For more information about policies, see Permissions and Policies in the IAM documentation.

Example 6: Permits a user to perform an action on a resource tagged with two different values

This following statement allows a user to perform the `ModifyDBInstance` and `CreateDBSnapshot` actions on instances with either the “stage” tag set to “development” or “test.”

```
{
"Version":"2012-10-17",
"Statement":[{
"Effect":"Allow",

"Action": [
```

```
    "rds:ModifyDBInstance",
    "rds:CreateDBSnapshot" ],
"Resource":"*",
"Condition":{"StringEquals":{"db-tag/stage": [ "development", "test" ] } }
  }]
}
```

Example 7:Permits a user to perform actions on a DB instance with a DB instance name prefixed with the user name

This following statement allows a user to perform any action (except to add or remove tags) on a DB instance that has a DB instance name that is prefixed with the user's name and that has a tag called "stage" equal to "devo" or that has no tag called "stage."

```
{
"Version":"2012-10-17",
"Statement":[{
"Effect":"Allow",
"NotAction": ["rds:AddTagsToResource","rds:RemoveTagsFromResource"],
"Resource": "arn:aws:rds:*:314195462963:db:${aws:username}*",
"Condition":{"StringEqualsIfExists":{"rds:db-tag/stage":"devo"}}
}]
}
```

Example 8: Prevent a user from deleting a DB instance

The following policy prevents a user from deleting a specific DB instance. For example, you may want to deny the ability to delete your production instances to any user that is not an administrator.

```
{
  "Version":"2012-10-17",
  "Statement":[{
    "Effect":"Deny",
    "Action":"rds:DeleteDBInstance",
    "Resource":"arn:aws:rds:us-east-1:314195462963:db:my-mysql-instance"
  }]
}
```

Supported Resource-Level Permissions for Amazon RDS API Actions

Resource-level permissions specify which resources users are allowed to perform actions on. For certain Amazon RDS actions, you can control when users are allowed to use those actions based on conditions that have to be fulfilled, or specific resources that users are allowed to use. For example, you can grant users permission to launch DB instances, but with only of a specific database class, and only using a specific option group.

The following example shows a policy that grants users permissions to create MySQL DB instances using the `mysql-production` parameter group.

```
{
"Version":"2012-10-17",
"Statement":[{
"Effect":"Allow",
"Action": "rds:CreateDBInstance",
"Resource":
    "arn:aws:rds:us-east-1:1234567890:pg:mysql-production",
    "Condition":{"StringEquals":{"rds:DatabaseEngine":"mysql"}}
  }]
}
```

The following table describes the Amazon RDS API actions that currently support resource-level permissions, as well as the supported resources (and their ARNs) and condition keys for each action.

Note
Not all of the Amazon RDS API actions support resource-level permissions. For a list of the API actions that do not support resource-level permissions, see Unsupported Amazon RDS Resource-Level Permissions (p. 106).

API Action	Resources	Condition Keys
AddSourceIdentifierToSubscription	Event subscription arn:aws:rds:*region*:*account*:es:*subscription-name*	rds:es-tag
AddTagsToResource	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag
	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
	DB security group arn:aws:rds:*region*:*account*:secgrp:*security-group-name*	rds:secgrp-tag
	DB subnet group arn:aws:rds:*region*:*account*:subgrp:*subnet-group-name*	rds:subgrp-tag
ApplyPendingMaintenanceAction	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag

API Action	Resources	Condition Keys
AuthorizeDBSecurityGroupIngress	DB security group arn:aws:rds:*region*:*account*:secgrp:*security-group-name*	rds:secgrp-tag
CopyDBParameterGroup	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
CopyDBSnapshot	DB snapshot arn:aws:rds:*region*:*account*:snapshot:*snapshot-name*	rds:snapshot-tag
CopyOptionGroup	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
CreateDBInstance	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:StorageSize rds:Vpc rds:db-tag
	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
	DB security group arn:aws:rds:*region*:*account*:secgrp:*security-group-name*	rds:secgrp-tag
	DB subnet group arn:aws:rds:*region*:*account*:subgrp:*subnet-group-name*	rds:subgrp-tag

API Action	Resources	Condition Keys
CreateDBInstanceReadReplica	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:DatabaseClass rds:Piops rds:db-tag
	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
	DB subnet group arn:aws:rds:*region*:*account*:subgrp:*subnet-group-name*	rds:subgrp-tag
CreateDBParameterGroup	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
CreateDBSecurityGroup	DB security group arn:aws:rds:*region*:*account*:secgrp:*security-group-name*	rds:secgrp-tag
CreateDBSnapshot	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag
	DB snapshot arn:aws:rds:*region*:*account*:snapshot:*snapshot-name*	rds:snapshot-tag
CreateDBSubnetGroup	DB subnet group arn:aws:rds:*region*:*account*:subgrp:*subnet-group-name*	rds:subgrp-tag
CreateEventSubscription	Event subscription arn:aws:rds:*region*:*account*:es:*subscription-name*	rds:es-tag
CreateOptionGroup	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
DeleteDBInstance	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag
	DB snapshot arn:aws:rds:*region*:*account*:snapshot:*snapshot-name*	rds:snapshot-tag

API Action	Resources	Condition Keys
DeleteDBParameterGroup	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
DeleteDBSecurityGroup	DB security group arn:aws:rds:*region*:*account*:secgrp:*security-group-name*	rds:secgrp-tag
DeleteDBSnapshot	DB snapshot arn:aws:rds:*region*:*account*:snapshot:*snapshot-name*	rds:snapshot-tag
DeleteDBSubnetGroup	DB subnet group arn:aws:rds:*region*:*account*:subgrp:*subnet-group-name*	rds:subgrp-tag
DeleteOptionGroup	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
DescribeDBEngineVersions	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
DescribeDBInstances	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag
DescribeDBLogFiles	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag
DescribeDBParameterGroups	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
DescribeDBParameters	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
DescribeDBSecurityGroups	DB security group arn:aws:rds:*region*:*account*:secgrp:*security-group-name*	rds:secgrp-tag

API Action	Resources	Condition Keys
DescribeDBSnapshots	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag
	DB snapshot arn:aws:rds:*region*:*account*:snapshot:*snapshot-name*	rds:snapshot-tag
DescribeDBSubnetGroups	DB subnet group arn:aws:rds:*region*:*account*:subgrp:*subnet-group-name*	rds:subgrp-tag
DescribeEvents	Event subscription arn:aws:rds:*region*:*account*:es:*subscription-name*	rds:es-tag
DeleteEventSubscription	Event subscription arn:aws:rds:*region*:*account*:es:*subscription-name*	rds:es-tag
DescribeEventSubscriptions	Event subscription arn:aws:rds:*region*:*account*:es:*subscription-name*	rds:es-tag
DescribeOptionGroupOptions	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
DescribeOptionGroups	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
DescribePendingMaintenanceActions	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:StorageSize rds:Vpc rds:db-tag

API Action	Resources	Condition Keys
DescribeReservedDBInstances	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:DatabaseClass rds:MultiAz rds:ri-tag
DescribeReservedDBInstancesOfferings	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:DatabaseClass rds:MultiAz
DownloadDBLogFilePortion	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag
ListTagsForResources	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag
	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
	DB security group arn:aws:rds:*region*:*account*:secgrp:*security-group-name*	rds:secgrp-tag
	DB subnet group arn:aws:rds:*region*:*account*:subgrp:*subnet-group-name*	rds:subgrp-tag

API Action	Resources	Condition Keys
ModifyDBInstance	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:DatabaseClass rds:MultiAz rds:Piops rds:StorageSize rds:Vpc rds:db-tag
	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
	DB security group arn:aws:rds:*region*:*account*:secgrp:*security-group-name*	rds:secgrp-tag
ModifyDBParameterGroup	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
ModifyDBSubnetGroup	DB subnet group arn:aws:rds:*region*:*account*:subgrp:*subnet-group-name*	rds:subgrp-tag
ModifyEventSubscription	Event subscription arn:aws:rds:*region*:*account*:es:*subscription-name*	rds:es-tag
ModifyOptionGroup	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
PromoteReadReplica	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag
RebootDBInstance	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag

API Action	Resources	Condition Keys
RemoveSourceIdentifierFromSubscription	Event subscription arn:aws:rds:*region*:*account*:es:*subscription-name*	rds:es-tag
RemoveTagsFromResource	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:db-tag
	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag
	DB security group arn:aws:rds:*region*:*account*:secgrp:*security-group-name*	rds:secgrp-tag
	DB subnet group arn:aws:rds:*region*:*account*:subgrp:*subnet-group-name*	rds:subgrp-tag
ResetDBParameterGroup	DB parameter group arn:aws:rds:*region*:*account*:pg:*parameter-group-name*	rds:pg-tag

API Action	Resources	Condition Keys
RestoreDBInstanceFromDBSnapshot	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:Vpc rds:db-tag
	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
	DB snapshot arn:aws:rds:*region*:*account*:snapshot:*snapshot-name*	rds:snapshot-tag
	DB subnet group arn:aws:rds:*region*:*account*:subgrp:*subnet-group-name*	rds:subgrp-tag
RestoreDBInstanceToPointInTime	DB instance arn:aws:rds:*region*:*account*:db:*db-instance-name*	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:Vpc rds:db-tag
	DB option group arn:aws:rds:*region*:*account*:og:*option-group-name*	rds:og-tag
	DB snapshot arn:aws:rds:*region*:*account*:snapshot:*snapshot-name*	rds:snapshot-tag
	DB subnet group arn:aws:rds:*region*:*account*:subgrp:*subnet-group-name*	rds:subgrp-tag

API Action	Resources	Condition Keys
RevokeDBSecurity-GroupIngress	DB security group arn:aws:rds:*region*:*account*:secgrp:*security-group-name*	rds:secgrp-tag

Unsupported Amazon RDS Resource-Level Permissions

The following Amazon RDS API actions do not support resource-level permissions:

- DescribeAccountAttributes
- DescribeCertificates
- DescribeEngineDefaultParameters
- DescribeEventCategories
- DescribeOrderableDBInstanceOptions
- PurchaseReservedDBInstancesOffering

Encrypting Amazon RDS Resources

You can encrypt your Amazon RDS instances and snapshots at rest by enabling the encryption option for your Amazon RDS DB instance. Data that is encrypted at rest includes the underlying storage for a DB instance, its automated backups, Read Replicas, and snapshots.

Amazon RDS encrypted instances use the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your Amazon RDS instance. Once your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently with a minimal impact on performance. You don't need to modify your database client applications to use encryption.

Amazon RDS encrypted instances provide an additional layer of data protection by securing your data from unauthorized access to the underlying storage. You can use Amazon RDS encryption to increase data protection of your applications deployed in the cloud, and to fulfill compliance requirements for data-at-rest encryption.

Amazon RDS encrypted instances are currently available for MySQL, PostgreSQL, Oracle, and SQL Server DB instances.

Amazon RDS also supports encrypting an Oracle or SQL Server DB instance with Transparent Data Encryption (TDE). TDE can be used in conjunction with encryption at rest, although using TDE and encryption at rest simultaneously might slightly affect the performance of your database. You must manage different keys for each encryption method. For more information on TDE, see Oracle Transparent Data Encryption (TDE) (p. 235), Appendix: Using AWS CloudHSM to Store Amazon RDS Oracle TDE Keys (p. 269), or SQL Server Transparent Data Encryption (p. 350).

To manage the keys used for encrypting and decrypting your Amazon RDS resources, you use the AWS Key Management Service (AWS KMS). AWS KMS combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Using AWS KMS, you can create encryption keys and define the policies that control how these keys can be used. AWS KMS supports CloudTrail, so you can audit key usage to verify that keys are being used appropriately. Your AWS KMS keys can be used in combination with Amazon RDS and supported AWS services such as Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS), and Amazon Redshift. For a list of services that support AWS KMS, go to Supported Services in the *AWS Key Management Service Developer Guide*.

All logs, backups, and snapshots are encrypted for an Amazon RDS encrypted instance. A Read Replica of an Amazon RDS encrypted instance is also encrypted using the same key as the master instance.

Enabling Amazon RDS Encryption for a DB Instance

To enable encryption for a new DB instance, select `Yes` in the **Enable encryption** dropdown in the Amazon RDS console. For information on creating a DB instance, see one of the following topics:

- Creating a DB Instance Running the MySQL Database Engine (p. 129)
- Creating a DB Instance Running the Oracle Database Engine (p. 205)
- Creating a DB Instance Running the SQL Server Database Engine (p. 313)
- Creating a DB Instance Running the PostgreSQL Database Engine (p. 363)

If you use the rds-create-db-instance CLI command to create an encrypted RDS DB instance, set the `--storage-encrypted` parameter to true. If you use the CreateDBInstance API action, set the `StorageEncrypted` parameter to true.

When you create an encrypted DB instance, you can also supply the AWS KMS key identifier for your encryption key. If you don't specify an AWS KMS key identifier, then Amazon RDS will use your default encryption key for your new DB instance. AWS KMS creates your default encryption key for Amazon RDS for your AWS account. Your AWS account has a different default encryption key for each AWS region.

Once you have created an encrypted DB instance, you cannot change the encryption key for that instance, Therefore, be sure to determine your encryption key requirements before you create your encrypted DB instance.

If you use the `rds-create-db-instance` CLI command to create an encrypted RDS DB instance, set the `--kms-key-id` parameter to the Amazon Resource Name (ARN) for the AWS KMS encryption key for the DB instance. If you use the `CreateDBInstance` API action, set the `KmsKeyId` parameter to the ARN for your AWS KMS key for the DB instance.

You can use the ARN of a key from another account to encrypt an RDS DB instance. If you create a DB instance with the same AWS account that owns the AWS KMS encryption key used to encrypt that new DB instance, the AWS KMS key ID that you pass can be the AWS KMS key alias instead of the key's ARN.

> **Important**
> If Amazon RDS loses access to the encryption key for a DB instance—for example, when Amazon RDS access to a key is revoked—then the encrypted DB instance is placed into a terminal state and can only be restored from a backup. We strongly recommend that you always enable backups for encrypted DB instances to guard against the loss of encrypted data in your databases.

Availability of Amazon RDS Encrypted Instances

Amazon RDS encrypted instances are currently available for MySQL, PostgreSQL, Oracle, and SQL Server DB instances.

Amazon RDS encryption is available for all storage types and the following DB instance classes:

Instance Type	Instance Class
General Purpose (M3)	db.m3.medium db.m3.large db.m3.xlarge db.m3.2xlarge
Memory Optimized (R3)	db.r3.large db.r3.xlarge db.r3.2xlarge db.r3.4xlarge db.r3.8xlarge
Memory Optimized—Previous Generation (CR1)	db.cr1.8xlarge

Note
Encryption at rest is not available for SQL Server Express Edition (sqlserver-ex) DB instances because sqlserver-ex is only available for the `db.t2.micro`, `db.t2.small`, and `db.t2.medium` DB instance classes.

Managing Amazon RDS Encryption Keys

You can manage keys used for Amazon RDS encrypted instances using the AWS Key Management Service (AWS KMS) in the IAM console. If you want full control over a key, then you must create a customer-managed key. You cannot delete, revoke, or rotate default keys provisioned by AWS KMS.

You can view audit logs of every action taken with a customer-managed key by using AWS CloudTrail.

Important
If you disable the key for an encrypted DB instance, you cannot read from or write to that DB instance. When Amazon RDS encounters a DB instance encrypted by a key that Amazon RDS does not have access to, Amazon RDS puts the DB instance into a terminal state where the DB instance is no longer available and the current state of the database cannot be recovered. In order to restore the DB instance, you must re-enable access to the encryption key for Amazon RDS, and then restore the DB instance from a backup.

Limitations of Amazon RDS Encrypted Instances

The following limitations exist for Amazon RDS encrypted instances:

- You can only enable encryption for an RDS DB instance when you create it, not after the DB instance is created.
- Existing DB instances that are not encrypted cannot be modified to enable encryption.
- DB instances that are encrypted cannot be modified to disable encryption.
- You cannot have an encrypted Read Replica of an unencrypted DB instance or an unencrypted Read Replica of an encrypted DB instance.
- Encrypted Read Replicas must be encrypted with the same key as the source DB instance.
- You cannot restore an unencrypted backup or snapshot to an encrypted DB instance.

- Because KMS encryption keys are specific to the region that they are created in, you cannot copy an encrypted snapshot from one region to another or replicate encrypted DB instances across regions.
- Because KMS encryption keys are specific to the region that they are created in, you cannot replicate encrypted DB instances across regions.

Using SSL to Encrypt a Connection to a DB Instance

You can use SSL from your application to encrypt a connection to a DB instance running MySQL, SQL Server, or PostgreSQL. Each DB engine has its own process for implementing SSL. To learn how to implement SSL for your DB instance, use the link following that corresponds to your DB engine:

- Using SSL with a MySQL DB Instance (p. 123)
- Using SSL with a SQL Server DB Instance (p. 310)
- Using SSL with a PostgreSQL DB Instance (p. 361)

A root certificate that works for all regions can be downloaded here. It is the trusted root entity and should work in most cases but might fail if your application does not accept certificate chains. If your application does not accept certificate chains, download the region-specific certificate from the following section.

SSL Certificate Rotation

Amazon RDS began updating the SSL certificates on all DB instances on March 23, 2015, but did not initiate a reboot of the instances. No operational impact or downtime is incurred when these updates are performed, and in many situations we will perform the update in your maintenance window. Amazon RDS will not update the certificate for your instances if you have already performed the update. Also note that Amazon RDS is not updating the certificates in AWS GovCloud (US) and the China (Beijing) regions.

Regardless of whether you manually update the certificate or Amazon RDS updated the certificate, the DB instance must be rebooted for the new certificate to take effect. You can decide when you want to manually reboot the DB instance, but you must update the certificate and reboot the instance before the old certificate (**rds-ca-2010**) expires on April 3, 2015.

You can check the certificate authority (CA) being used by your DB instance using the Amazon RDS console. The CA is listed under the **Security and Network** section of your DB instance details. If your instance shows **rds-ca-2015**, then the new certificate has been successfully applied. You still need to reboot your database instance and update your client application to use the new SSL certificate.

If the Amazon RDS console shows your instance CA as **rds-ca-2010**, then the new certificate has not been applied to your database instance yet. Use the instructions following to update the SSL certificate on your database instances.

Several types of certificates are available. Select the appropriate certificate for your client.

We encourage you to test these steps on a development or staging environment before applying them on your production environment. Note that when you complete step 4, your DB instance will be rebooted.

To apply a new SSL certificate to your DB instance

1. Select the certificate(s) you need.

   - A root certificate that works for all regions can be downloaded here. It is the trusted root entity and should work in most cases but might fail if your application does not accept certificate chains.

- If you need an intermediate certificate for a particular region, download the certificate by selecting the region the DB instance resides in from the following list:

  Asia Pacific (Tokyo)

  Asia Pacific (Singapore)

  Asia Pacific (Sydney)

  EU (Frankfurt)

  EU (Ireland)

  South America (Sao Paulo)

  US East (N. Virginia)

  US West (N. California)

  US West (Oregon)
- A certificate bundle that contains both the old and new root certificates can be downloaded here.
- If your application is on the Microsoft Windows platform and requires a PKCS7 file, you can download the PKCS7 certificate bundle that contains both the old and new certificates here.

2. Modify the configuration file of your database client or application to use the certificate you selected in step 1.
3. In the Amazon RDS console, click the DB instance, and then click **Modify** under **Instance Actions**.
4. In the **Modify DB Instance** dialog box, in the **Certificate Authority** text box, change the certificate from *rds-ca-2010* to *rds-ca-2015*. Click **Apply Immediately** so that the change to the SSL certificate for the DB instance is applied immediately. Click **Continue**. Review the change and click **Modify DB Instance**. This action will rotate the SSL certificate on the DB instance and initiate a reboot operation to make the certificate take effect.

   Important
   This step will cause your DB instance to reboot. If you do not want your DB instance to reboot immediately when you click **Modify DB Instance**, postpone this step until it is safe to reboot the DB instance.

The database reboot operation typically takes less than two minutes to complete. In some cases, such as when the database has a large number of tables, the operation can take longer. For more information about rebooting a DB instance, see Rebooting a DB Instance (p. 462).

If you don't use SSL to encrypt your connection, you don't have to take any action. The certificate for your DB instance will be updated after March 23, 2015, without a reboot. The certificate change will not affect the instance's performance. If in the future you decide to use SSL encryption, you will have to reboot the DB instance for the certificate to take effect.

Amazon RDS Security Groups

Security groups control the access that traffic has in and out of a DB instance. Three types of security groups are used with Amazon RDS: DB security groups, VPC security groups, and EC2 security groups. In simple terms, a DB security group controls access to a DB instance that is not in a VPC, a VPC security group controls access to a DB instance (or other AWS instances) inside a VPC, and an EC2 security group controls access to an EC2 instance.

By default, network access is turned off to a DB instance. You can specify rules in a security group that allows access from an IP address range, port, or EC2 security group. Once ingress rules are configured, the same rules apply to all DB instances that are associated with that security group. You can specify up to 20 rules in a security group.

DB Security Groups

Each DB security group rule enables a specific source to access a DB instance that is associated with that DB security group. The source can be a range of addresses (e.g., 203.0.113.0/24), or an EC2 security group. When you specify an EC2 security group as the source, you allow incoming traffic from all EC2 instances that use that EC2 security group. Note that DB security group rules apply to inbound traffic only; outbound traffic is not currently permitted for DB instances.

You do not need to specify a destination port number when you create DB security group rules; the port number defined for the DB instance is used as the destination port number for all rules defined for the DB security group. DB security groups can be created using the Amazon RDS APIs or the Amazon RDS page of the AWS Management Console.

For more information about working with DB security groups, see Working with DB Security Groups (p. 537)

VPC Security Groups

Each VPC security group rule enables a specific source to access a DB instance in a VPC that is associated with that VPC security group. The source can be a range of addresses (e.g., 203.0.113.0/24), or another VPC security group. By specifying a VPC security group as the source, you allow incoming traffic from all instances (typically application servers) that use the source VPC security group. VPC security groups can have rules that govern both inbound and outbound traffic, though the outbound traffic rules do not apply to DB instances. Note that you must use the Amazon EC2 API or the **Security Group** option on the VPC Console to create VPC security groups.

DB instances deployed within a VPC can be configured to be accessible from the Internet or from EC2 instances outside the VPC. If a VPC security group specifies a port access such as TCP port 22, you would not be able to access the DB instance because the firewall for the DB instance provides access only via the IP addresses specified by the DB security groups the instance is a member of and the port defined when the DB instance was created.

You should use TCP as the protocol for any VPC security group created to control access to a DB instance. The port number for the VPC security group should be the same port number as that used to create the DB instance.

DB Security Groups vs. VPC Security Groups

The following table shows the key differences between DB security groups and VPC security groups.

DB Security Group	VPC Security Group
Controls access to DB instances outside a VPC	Controls access to DB instances in VPC.
Uses Amazon RDS APIs or Amazon RDS page of the AWS Management Console to create and manage group/rules	Uses Amazon EC2 APIs or Amazon VPC page of the AWS Management Console to create and manage group/rules.
When you add a rule to a group, you do not need to specify port number or protocol.	When you add a rule to a group, you should specify the protocol as TCP, and specify the same port number that you used to create the DB instances (or Options) you plan to add as members to the group.

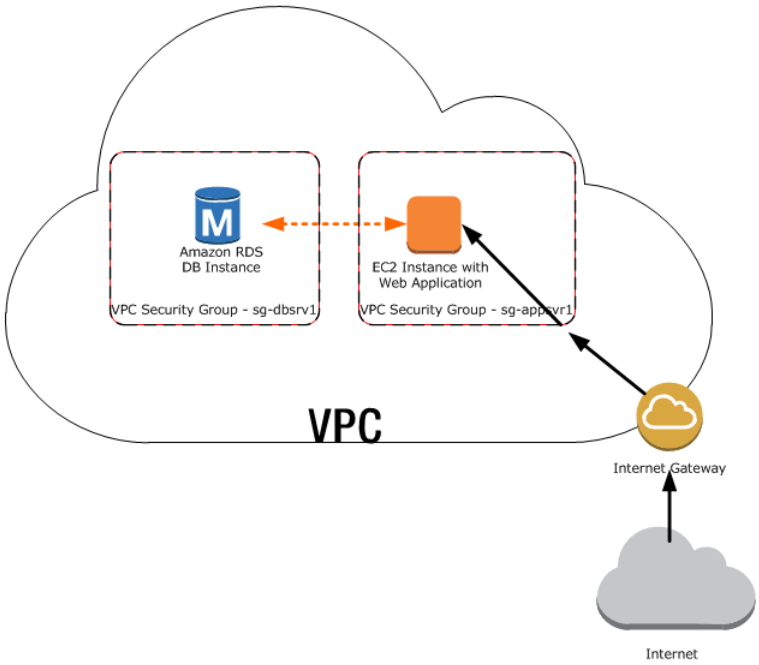
DB Security Group	VPC Security Group
Groups allow access from EC2 security groups in your AWS account or other accounts.	Groups allow access from other VPC security groups in your VPC only.

Security Group Scenario

A common use of an RDS instance in a VPC is to share data with an application server running in an EC2 instance in the same VPC and that is accessed by a client application outside the VPC. For this scenario, you would do the following to create the necessary instances and security groups. You can use the RDS and VPC pages on the AWS Console or the RDS and EC2 APIs.

1. Create a VPC security group (for example, "sg-appsrv1") and define inbound rules that use as source the IP addresses of the client application. This security group allows your client application to connect to EC2 instances in a VPC that uses this security group.
2. Create an EC2 instance for the application and add the EC2 instance to the VPC security group ("sg-appsrv1") you created in the previous step. The EC2 instance in the VPC shares the VPC security group with the DB instance.
3. Create a second VPC security group (for example, "sg-dbsrv1") and create a new rule by specifying the VPC security group you created in step 1 ("sg-appsrv1") as the source.
4. Create a new DB instance and add the DB instance to the VPC security group ("sg-dbsrv1") you created in the previous step. When you create the instance, use the same port number as the one specified for the VPC security group ("sg-dbsrv1") rule you created in step 3.

The following diagram shows this scenario.



For more information on working with DB security groups, go to Working with DB Security Groups (p. 537).

Related Topics

- Working with DB Security Groups (p. 537)

Limits for Amazon RDS

This topic describes the resource limits and naming constraints for Amazon RDS.

Topics

- Limits in Amazon RDS (p. 114)
- Naming Constraints in Amazon RDS (p. 115)
- File Size Limits in Amazon RDS (p. 117)

Limits in Amazon RDS

Each AWS account has limits, per region, on the number of Amazon RDS resources that can be created. Once a limit for a resource has been reached, additional calls to create that resource will fail with an exception.

The following table lists the resources and their limits per region.

Resource	Default Limit
Instances	40
Reserved instances	40
Total storage for all DB instances	100 TB
Manual snapshots	50
Parameter groups	50
Security groups	25
Subnet groups	20
Subnets per subnet group	20
Option groups	20
Event subscriptions	20
Read replicas per master	5

Naming Constraints in Amazon RDS

The following table describes naming constraints in Amazon RDS.

DB instance identifier	• Must contain from 1 to 63 alphanumeric characters or hyphens (1 to 15 for SQL Server). • First character must be a letter. • Cannot end with a hyphen or contain two consecutive hyphens. • Must be unique for all DB instances per AWS account, per region.
Database name	Database name constraints differ for each database engine. **MySQL** • Must contain 1 to 64 alphanumeric characters. • Cannot be a word reserved by the database engine. **PostgreSQL** • Must contain 1 to 63 alphanumeric characters. • Must begin with a letter or an underscore. Subsequent characters can be letters, underscores, or digits (0-9). • Cannot be a word reserved by the database engine. **Oracle** • Cannot be longer than 8 characters. **SQL Server** • Not applicable.

Master user name	Master user name constraints differ for each database engine. **MySQL** • Must contain 1 to 16 alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine. **Oracle** • Must contain 1 to 30 alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine. **SQL Server** • Must contain 1 to 128 alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine. **PostgreSQL** • Must contain 1 to 63 alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine.
Master password	The password for the master database user can be any printable ASCII character except "/", """, or "@". Master password constraints differ for each database engine. **MySQL** • Must contain 8 to 41 characters. **Oracle** • Must contain 8 to 30 characters. **SQL Server** • Must contain 8 to 128 characters. **PostgreSQL** • Must contain 8 to 128 characters .

DB parameter group name	• Must contain from 1 to 255 alphanumeric characters. • First character must be a letter. • Cannot end with a hyphen or contain two consecutive hyphens.

File Size Limits in Amazon RDS

Amazon RDS instances can support files with a maximum size of 2 TB due to underlying file system constraints.

MySQL File Size Limits in Amazon RDS

With MySQL, this file size limit constrains each table to a maximum size of 2 TB when using InnoDB file-per-table tablespaces. This limit also constrains the system tablespace to a maximum size of 2 TB. File-per-table tablespaces with tables each in their own tablespace is set by default in MySQL version 5.6.6 and later. You must enable InnoDB file-per-table tablespaces for MySQL versions 5.1 and 5.5.

There are advantages and disadvantages to using InnoDB file-per-table tablespaces, depending on your application. To determine the best approach for your application, go to InnoDB File-Per-Table Mode in the MySQL documentation.

We don't recommend allowing tables to grow to 2 TB. In general, a better practice is to partition data into smaller tables, which can improve performance and recovery times.

One option that you can use for breaking a large table up into smaller tables is partitioning. Partitioning distributes portions of your large table into separate files based on rules that you specify. For example, if you store transactions by date, you can create partitioning rules that distribute older transactions into separate files using partitioning. Then periodically, you can archive the historical transaction data that doesn't need to be readily available to your application. For more information, go to https://dev.mysql.com/doc/refman/5.6/en/partitioning.htmlin the MySQL documentation.

To determine the file size of a table

Use the following SQL command to determine if any of your tables are too large and are candidates for partitioning.

```
SELECT TABLE_SCHEMA, TABLE_NAME,
       round(((DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024), 2) As "Approximate
size (MB)"
FROM information_schema.TABLES
WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema');
```

To enable InnoDB file-per-table tablespaces

- To enable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to 1 in the parameter group for the DB instance.

To disable InnoDB file-per-table tablespaces

- To disable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to `0` in the parameter group for the DB instance.

For information on updating a parameter group, see Working with DB Parameter Groups (p. 523).

When you have enabled or disabled InnoDB file-per-table tablespaces, you can issue an `ALTER TABLE` command to move a table from the global tablespace to its own tablespace, or from its own tablespace to the global tablespace as shown in the following example:

```
ALTER TABLE table_name ENGINE=InnoDB;
```

MySQL on Amazon RDS

Amazon RDS supports DB instances running several versions of MySQL. You first use the Amazon RDS management tools or interfaces to create an Amazon RDS MySQL DB instance. You can then use the Amazon RDS tools to perform management actions for the DB instance, such as reconfiguring or resizing the DB instance, authorizing connections to the DB instance, creating and restoring from backups or snapshots, creating Multi-AZ secondaries, creating Read Replicas, and monitoring the performance of the DB instance. You use standard MySQL utilities and applications to store and access the data in the DB instance.

These are the common management tasks you perform with an Amazon RDS MySQL DB instance, with links to information about each task:

- For planning information, such as MySQL versions, storage engines, security, and features supported in Amazon RDS, see MySQL on Amazon RDS Planning Information (p. 120).
- Before creating a DB instance, you should complete the steps in the Setting Up for Amazon RDS (p. 7) section of this guide.
- You can create an Amazon RDS MySQL DB instance after you have met prerequisites, such as creating security groups, DB parameter groups, or DB option groups. For information, see Creating a DB Instance Running the MySQL Database Engine (p. 129).
- After creating the security group and DB instance, you can connect to the DB instance from MySQL applications and utilities. For information, see Connecting to a DB Instance Running the MySQL Database Engine (p. 137).
- A newly created Amazon RDS DB instance has one empty database with the name you specified when you created the DB instance, and one masteruser account with the name and password you specified. You must use a MySQL tool or utility to log in as the masteruser, and then use MySQL commands and SQL statements to add all of the users and elements required for your applications to store and retrieve data in the DB instance, such as:
  - Create all user IDs and grant them the appropriate permissions. For information, go to MySQL User Account Management in the MySQL documentation.
  - Create any required databases and objects such as tables and views. For information, go to Data Definition Statements in the MySQL documentation.
  - Establish procedures for importing or exporting data. For information on some recommended procedures, see Importing and Exporting Data From a MySQL DB Instance (p. 143).
- You may need to periodically change your DB instance, such as to resize or reconfigure the DB instance. For information, see Modifying a DB Instance Running the MySQL Database Engine (p. 140). For additional information on specific tasks, see:
  - Renaming a DB Instance (p. 457)

  - Deleting a DB Instance (p. 459)
  - Rebooting a DB Instance (p. 462)
  - Tagging Amazon RDS Resources (p. 486)
  - Major DB Engine Version Upgrades for a DB Instance (p. 447)
  - Adjusting the Preferred Maintenance Window (p. 453)
- You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots. For information, see Backing Up and Restoring (p. 495).
- You can monitor an instance through actions such as viewing the MySQL logs, CloudWatch Amazon RDS metrics, and events. For information, see Monitoring Amazon RDS (p. 571).
- You can offload read traffic from your primary MySQL DB instance by creating Read Replicas. For information, see Working with PostgreSQL and MySQL Read Replicas (p. 472).
- There are several Amazon RDS features you can use with MySQL DB instances that are common across the Amazon RDS database engines. For information, see:
  - Working with Reserved DB Instances (p. 550)
  - Amazon RDS Provisioned IOPS Storage to Improve Performance (p. 82)

There are also several appendices with useful information about working with Amazon RDS MySQL DB instances:

- Appendix: Common DBA Tasks for MySQL (p. 168)
- Appendix: Options for MySQL Database Engine (p. 173)
- Appendix: MySQL on Amazon RDS SQL Reference (p. 177)

MySQL on Amazon RDS Planning Information

Topics

- MySQL on Amazon RDS Versions (p. 120)
- Amazon RDS Supported Storage Engines (p. 121)
- Amazon RDS and MySQL Security (p. 122)
- InnoDB Cache Warming (p. 123)
- MySQL Features Not Supported By Amazon RDS (p. 124)
- Known Issues and Limitations (p. 125)

MySQL on Amazon RDS Versions

Amazon RDS currently supports MySQL versions 5.6, 5.5, and 5.1. Over time, we plan to support additional MySQL versions for Amazon RDS. The number of new version releases supported in a given year will vary based on the frequency and content of the MySQL version releases and the outcome of a thorough vetting of the release by our database engineering team. However, as a general guidance, we aim to support new MySQL versions within 3-5 months of their General Availability release.

MySQL, version numbers are organized as version = X.Y.Z. In Amazon RDS terminology, X.Y denotes the major version, and Z is the minor version number. For Amazon RDS implementations, a version change would be considered major if the major version number changes; for example, going from version 5.1.71 to 5.5.33. A version change would be considered minor if only the minor version number changes - for example, going from version 5.5.31 to 5.5.33.

You can specify any currently supported MySQL version when creating a new DB Instance. You can specify the MySQL 5.6, 5.5, or 5.1 major versions, and any supported minor version for the specified

major version. If no version is specified, Amazon RDS will default to a supported version, typically the most recent version. If a major version (e.g. MySQL 5.6) is specified but a minor version is not, Amazon RDS will default to a recent release of the major version you have specified. To see a list of supported versions, as well as defaults for newly created DB Instances, use the DescribeDBEngineVersions API.

With Amazon RDS, you control when to upgrade your MySQL instance to a new version supported by Amazon RDS. You can maintain compatibility with specific MySQL versions, test new versions with your application before deploying in production, and perform version upgrades at times that best fit your schedule.

Unless you specify otherwise, your DB Instance will automatically be upgraded to new MySQL minor versions as they are supported by Amazon RDS. This patching will occur during your scheduled maintenance window, and it will be announced on the Amazon RDS Community Forum in advance. To turn off automatic version upgrades, set the AutoMinorVersionUpgrade parameter to "false."

If you opt out of automatically scheduled upgrades, you can manually upgrade to a supported minor version release by following the same procedure as you would for a major version update. For information, see Major DB Engine Version Upgrades for a DB Instance (p. 447).

Amazon RDS currently supports the major version upgrades from MySQL version 5.1 to version 5.5 and from MySQL version 5.5 to version 5.6. Because major version upgrades involve some compatibility risk, they will not occur automatically; you must make a request to modify the DB instance. You should thoroughly test any upgrade before upgrading your production instances. For information about upgrading a DB instance, see Major DB Engine Version Upgrades for a DB Instance (p. 447).

You can test a DB Instance against a new version before upgrading by creating a DB Snapshot of your existing DB Instance, restoring from the DB Snapshot to create a new DB Instance, and then initiating a version upgrade for the new DB Instance. You can then experiment safely on the upgraded clone of your DB Instance before deciding whether or not to upgrade your original DB Instance.

The Amazon RDS deprecation policy for MySQL includes the following:

- We intend to support major MySQL version releases, including MySQL 5.1, for 3 years after they are initially supported by Amazon RDS.
- We intend to support minor MySQL version releases (e.g. MySQL 5.1.45) for at least 1 year after they are initially supported by Amazon RDS.
- After a MySQL major or minor version has been "deprecated", we expect to provide a three month grace period for you to initiate an upgrade to a supported version prior to an automatic upgrade being applied during your scheduled maintenance window.

Using the memcached Option with MySQL 5.6

Most Amazon RDS DB engines support option groups that allow you to select additional features for your DB instance. MySQL 5.6 DB instances support the `memcached` option, a simple, key-based cache. For more information about the `memcached` option, see Appendix: Options for MySQL Database Engine (p. 173). For more information about working with option groups, see Working with Option Groups (p. 510).

Amazon RDS Supported Storage Engines

While MySQL supports multiple storage engines with varying capabilities, not all of them are optimized for recovery and data durability. Amazon RDS fully supports the InnoDB storage engine for MySQL DB instances. Amazon RDS features such as Point-In-Time restore and snapshot restore require a recoverable storage engine and are supported for the InnoDB storage engine only. You must be running an instance of MySQL 5.6 to use the InnoDB `memcached` interface. For more information, see MySQL 5.6 memcached Support (p. 173).

The Federated Storage Engine is currently not supported by Amazon RDS for MySQL.

The MyISAM storage engine does not support reliable recovery and may result in lost or corrupt data when MySQL is restarted after a recovery, preventing Point-In-Time restore or snapshot restore from working as intended. However, if you still choose to use MyISAM with Amazon RDS, snapshots may be helpful under some conditions. For more information on MyISAM restrictions, see Automated Backups with Unsupported MySQL Storage Engines (p. 75).

If you would like to convert existing MyISAM tables to InnoDB tables, you can use the alter table command (e.g., alter table TABLE_NAME engine=innodb;). Please bear in mind that MyISAM and InnoDB have different strengths and weaknesses, so you should fully evaluate the impact of making this switch on your applications before doing so.

Amazon RDS and MySQL Security

Security for Amazon RDS MySQL DB instances is managed at three levels:

- AWS Identity and Access Management controls who can perform Amazon RDS management actions on DB instances. When you connect to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS management operations. For more information, see Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources (p. 88).
- When you create a DB instance, you use either a VPC security group or a DB security group to control which devices and Amazon EC2 instances can open connections to the endpoint and port of the DB instance. These connections can be made using SSL. In addition, firewall rules at your company can control whether devices running at your company can open connections to the DB instance.
- Once a connection has been opened to a MySQL DB instance, authentication of the login and permissions are applied the same way as in a stand-alone instance of MySQL. Commands such as `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE`, and `SET PASSWORD` work just as they do in stand-alone databases, as does directly modifying database schema tables. For information, go to MySQL User Account Management in the MySQL documentation.

When you create an Amazon RDS DB instance, the master user has the following default privileges:

- `alter`
- `alter routine`
- `create`
- `create routine`
- `create temporary tables`
- `create user`
- `create view`
- `delete`
- `drop`
- `event`
- `execute`
- `grant option`
- `index`
- `insert`
- `lock tables`
- `process`
- `references`
- `replication slave`
- `select`

- `show databases`
- `show view`
- `trigger`
- `update`

> **Note**
> Although it is possible to delete the master user on the DB instance, it is not recommended. To recreate the master user, use the `ModifyDBInstance` API or the `rds-modify-db-instance` command line tool and specify a new master user password with the appropriate parameter. If the master user does not exist in the instance, the master user will be created with the specified password.

To provide management services for each DB instance, the `rdsadmin` user is created when the DB instance is created. Attempting to drop, rename, change the password, or change privileges for the `rdsadmin` account will result in an error.

To allow management of the DB instance, the standard `kill` and `kill_query` commands have been restricted. The Amazon RDS commands `rds_kill` and `rds_kill_query` are provided to allow you to terminate user sessions or queries on DB instances.

Using SSL with a MySQL DB Instance

Amazon RDS supports SSL connections with DB instances running the MySQL database engine.

Amazon RDS creates an SSL certificate and installs the certificate on the DB instance when Amazon RDS provisions the instance. These certificates are signed by a certificate authority. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. The public key is stored at http://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem.

> **Important**
> Amazon RDS will rotate all SSL certificates for DB instances on March 23, 2015 but will not initiate a reboot of the instance. If you use SSL to connect to an Amazon RDS DB instance, you must follow the steps in the topic SSL Certificate Rotation (p. 109) to apply a new SSL certificate to your DB instance before March 23, 2015 or you will not be able to connect to the DB instance using SSL.

To encrypt connections using the default **mysql** client, launch the mysql client using the *`--ssl-ca parameter`* to reference the public key, for example:

```
mysql -h myinstance.c9akciq32.rds-cn-north-1.amazonaws.com
--ssl-ca=rds-ssl-ca-cert.pem --ssl-verify-server-cert
```

You can use the GRANT statement to require SSL connections for specific users accounts. For example, you can use the following statement to require SSL connections on the user account encrypted_user:

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL
```

> **Note**
> For more information on SSL connections with MySQL, go to the MySQL documentation.

InnoDB Cache Warming

InnoDB cache warming can provide performance gains for your MySQL DB instance by saving the current state of the buffer pool when the DB instance is shut down, and then reloading the buffer pool from the saved information when the DB instance starts up. This bypasses the need for the buffer pool to "warm up" from normal database use and instead preloads the buffer pool with the pages for known common

queries. The file that stores the saved buffer pool information only stores metadata for the pages that are in the buffer pool, and not the pages themselves. As a result, the file does not require much storage space. The file size is about 0.2 percent of the cache size. For example, for a 64 GB cache, the cache warming file size is 128 MB. For more information on InnoDB cache warming, go to Preloading the InnoDB Buffer Pool for Faster Restart in the MySQL documentation.

MySQL on Amazon RDS supports InnoDB cache warming for MySQL version 5.6 and later. To enable InnoDB cache warming, set the `innodb_buffer_pool_dump_at_shutdown` and `innodb_buffer_pool_load_at_startup` parameters to 1 in the parameter group for your DB instance. Changing these parameter values in a parameter group will affect all MySQL DB instances that use that parameter group. To enable InnoDB cache warming for specific MySQL DB instances, you might need to create a new parameter group for those instances. For information on parameter groups, see Working with DB Parameter Groups (p. 523).

InnoDB cache warming primarily provides a performance benefit for DB instances that use standard storage. If you use PIOPS storage, you do not commonly see a significant performance benefit.

Important
If your MySQL DB instance does not shut down normally, such as during a failover, then the buffer pool state will not be saved to disk. In this case, MySQL loads whatever buffer pool file is available when the DB instance is restarted. No harm is done, but the restored buffer pool might not reflect the most recent state of the buffer pool prior to the restart. To ensure that you have a recent state of the buffer pool available to warm the InnoDB cache on startup, we recommend that you periodically dump the buffer pool "on demand." You can dump or load the buffer pool on demand if your DB instance is running MySQL version 5.6.19 or later.
You can create an event to dump the buffer pool automatically and on a regular interval. For example, the following statement creates an event named `periodic_buffer_pool_dump` that dumps the buffer pool every hour.

```
CREATE EVENT periodic_buffer_pool_dump ON SCHEDULE EVERY 1 HOUR DO CALL
 mysql.rds_innodb_buffer_pool_dump_now();
```

For more information on MySQL events, see Event Syntax in the MySQL documentation.

Dumping and Loading the Buffer Pool on Demand

For MySQL version 5.6.19 and later, you can save and load the InnoDB cache "on demand."

- To dump the current state of the buffer pool to disk, call the mysql.rds_innodb_buffer_pool_dump_now (p. 184) stored procedure.
- To load the saved state of the buffer pool from disk, call the mysql.rds_innodb_buffer_pool_load_now (p. 185) stored procedure.
- To cancel a load operation in progress, call the mysql.rds_innodb_buffer_pool_load_abort (p. 185) stored procedure.

MySQL Features Not Supported By Amazon RDS

Amazon RDS currently does not support the following MySQL features:

- Global Transaction IDs
- Transportable Table Space
- Authentication Plugin
- Password Strength Plugin

- Semi-synchronous Replication

In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application. Amazon RDS does not allow direct host access to a DB instance via Telnet, Secure Shell (SSH), or Windows Remote Desktop Connection. When you create a DB instance, you are assigned to the *db_owner* role for all databases on that instance, and you will have all database-level permissions except for those used for backups (Amazon RDS manages backups for you).

Known Issues and Limitations

Memcached recommended MySQL version

We recommend that you only use the `memcached` interface with MySQL version 5.6.21b or later. This is because there are a number of bug fixes related to the `memcached` interface which are included in the MySQL engine starting with version 5.6.21b. For more information, go to Changes in MySQL 5.6.20 (2014-07-31) and Changes in MySQL 5.6.21 (2014-09-23) in the MySQL documentation.

For more information on using `memcached` with MySQL on Amazon RDS, see MySQL 5.6 memcached Support (p. 173)

MySQL Version 5.5.40 Asynchronous I/O Is Disabled

You might observe reduced I/O performance if you have a MySQL DB instance that was created before April 23, 2014 and then upgraded to MySQL version 5.5.40 after October 17, 2014. This reduced performance can be caused by an error that disables the `innodb_use_native_aio` parameter even if the corresponding DB parameter group enables the `innodb_use_native_aio` parameter.

To resolve this error, we recommend that you upgrade your MySQL DB instance running version 5.5.40 to version 5.5.40a, which corrects this behavior. For information on minor version upgrades, see Minor DB Engine Version Upgrades for a DB Instance (p. 447).

For more information on MySQL asynchronous I/O, go to Asynchronous I/O on Linux in the MySQL documentation.

Index Merge Optimization Returns Wrong Results

Queries that use index merge optimization might return wrong results due to a bug in the MySQL query optimizer that was introduced in MySQL 5.5.37. When you issue a query against a table with multiple indexes the optimizer scans ranges of rows based on the multiple indexes, but does not merge the results together correctly. For more information on the query optimizer bug, go to http://bugs.mysql.com/bug.php?id=72745 and http://bugs.mysql.com/bug.php?id=68194 in the MySQL bug database.

For example, consider a query on a table with two indexes where the search arguments reference the indexed columns.

```
SELECT * FROM table1
  WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

In this case, the search engine will search both indexes. However, due to the bug, the merged results will be incorrect.

To resolve this issue, you can do one of the following:

- Set the `optimizer_switch` parameter to `index_merge=off` in the DB parameter group for your MySQL DB instance. For information on setting DB parameter group parameters, see Working with DB Parameter Groups (p. 523).
- Upgrade your MySQL DB instance to MySQL version 5.6.19a. For information on major version upgrades, see Major DB Engine Version Upgrades for a DB Instance (p. 447).
- If you cannot upgrade your instance or change the `optimizer_switch` parameter, you can work around the bug by explicitly identifying an index for the query, for example:

```
SELECT * FROM table1
  USE INDEX covering_index
  WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

For more information, go to Index Merge Optimization.

Replication Fails After Upgrading to MySQL Version 5.6.21

If you have a DB instance that runs a version prior to version 5.6.4, or if the DB instance was upgraded from a version prior to version 5.6.4, you can receive the following error if you have a Read Replica that runs MySQL version 5.6.21.

```
mysqld got signal 11 ;
This could be because you hit a bug. It is also possible that this binary
or one of the libraries it was linked against is corrupt, improperly built,
or misconfigured. This error can also be caused by malfunctioning hardware.
We will try our best to scrape up some info that will hopefully help
diagnose the problem, but since we have already crashed,
something is definitely wrong and this may fail.
```

MySQL version 5.6.4 introduced a new date and time format for `datetime`, `time`, and `timestamp` columns that allows fractional components in date and time values. The error is caused by a mismatch in date and time formats between the master and the replica, and results in a failure when row-based logging attempts to replay an operation from the master DB instance to the replica DB instance. You might also see a number of related row-based logging messages in your MySQL error log, for example: `Relay_log_info`, `Rows_log_event`, and so on. For information on the new date and time format for MySQL, go to Upgrading from MySQL 5.5 to 5.6 in the MySQL documentation..

To resolve the error, you can do either of the following:

- Upgrade your Read Replica to MySQL version 5.6.23 or later. For information on upgrading a MySQL DB instance on Amazon RDS to version 5.6, see Upgrading Database Versions for a DB Instance (p. 446).
- Upgrade your master DB instance to MySQL version 5.6.12 or later and update the format of the affected date and time columns. For information on upgrading a MySQL DB instance on Amazon RDS to version 5.6, see Upgrading Database Versions for a DB Instance (p. 446).

  To upgrade your date and time columns to the new format on your master DB instance, you must issue the `ALTER TABLE <table_name> FORCE;` command.

Note
Because altering a table locks the table as read-only, we recommend that you perform this update during a maintenance window.

You can run the following query to find all of the tables in your database that have columns of type `datetime`, `time`, or `timestamp` and create an `ALTER TABLE <table_name> FORCE;` command for each table.

```
SELECT DISTINCT CONCAT('ALTER TABLE `',
  REPLACE(is_tables.TABLE_SCHEMA, '`', '``'), '`.`',
  REPLACE(is_tables.TABLE_NAME, '`', '``'), '` FORCE;')
FROM information_schema.TABLES is_tables
  INNER JOIN information_schema.COLUMNS col ON col.TABLE_SCHEMA =
is_tables.TABLE_SCHEMA
    AND col.TABLE_NAME = is_tables.TABLE_NAME
  LEFT OUTER JOIN information_schema.INNODB_SYS_TABLES systables ON
    SUBSTRING_INDEX(systables.NAME, '#', 1) = CON
CAT(is_tables.TABLE_SCHEMA,'/',is_tables.TABLE_NAME)
  LEFT OUTER JOIN information_schema.INNODB_SYS_COLUMNS syscolumns ON
    syscolumns.TABLE_ID = systables.TABLE_ID AND syscolumns.NAME =
col.COLUMN_NAME
WHERE col.COLUMN_TYPE IN ('time','timestamp','datetime')
  AND is_tables.TABLE_TYPE = 'BASE TABLE'
  AND is_tables.TABLE_SCHEMA NOT IN ('mysql','information_schema','perform
ance_schema')
  AND (is_tables.ENGINE = 'InnoDB' AND syscolumns.MTYPE = 6);
```

Log File Size

For MySQL version 5.6.20 and later, there is a size limit on BLOBs written to the redo log. To account for this limit, ensure that the `innodb_log_file_size` parameter for your MySQL DB instance is 10 times larger than the largest BLOB data size found in your tables, plus the length of other variable length fields (`VARCHAR`, `VARBINARY`, `TEXT`) in the same tables. For information on how to set parameter values, see Working with DB Parameter Groups (p. 523). For information on the redo log BLOB size limit, go to Changes in MySQL 5.6.20.

MySQL Parameter Exceptions for Amazon RDS DB Instances

Some MySQL parameters require special considerations when used with an Amazon RDS DB instance.

lower_case_table_names

Because Amazon RDS uses a case-sensitive file system, setting the value of the `lower_case_table_names` server parameter to 2 ("names stored as given but compared in lowercase") is not supported. Supported values for Amazon RDS DB Instances are 0 ("names stored as given and comparisons are case-sensitive"), which is the default, or 1 ("names stored in lowercase and comparisons are not case-sensitive").

The `lower_case_table_names` parameter should be set as part of a custom DB parameter group before creating a DB instance. You should avoid changing the *lower_case_table_names* parameter for existing database instances because doing so could cause inconsistencies with point-in-time recovery backups and Read Replica DB instances.

Read Replicas should always use the same *lower_case_table_names* parameter value as the master DB Instance.

long_query_time

You can set the `long_query_time` parameter to a floating point value which allows you to log slow queries to the MySQL slow query log with microsecond resolution. You can set a value such as 0.1 seconds, which would be 100 milliseconds, to help when debugging slow transactions that take less than one second.

MySQL File Size Limits

Amazon RDS instances can support files with a maximum size of 2 TB due to underlying file system constraints.

With MySQL, this file size limit constrains each table to a maximum size of 2 TB when using InnoDB file-per-table tablespaces. This limit also constrains the system tablespace to a maximum size of 2 TB. File-per-table tablespaces with tables each in their own tablespace is set by default in MySQL version 5.6.6 and later. You must enable InnoDB file-per-table tablespaces for MySQL versions 5.1 and 5.5.

There are advantages and disadvantages to using InnoDB file-per-table tablespaces, depending on your application. To determine the best approach for your application, go to InnoDB File-Per-Table Mode in the MySQL documentation.

We don't recommend allowing tables to grow to 2 TB. In general, abetter practice is to partition data into smaller tables, which can improve performance and recovery times.

One option that you can use for breaking a large table up into smaller tables is partitioning. Partitioning distributes portions of your large table into separate files based on rules that you specify. For example, if you store transactions by date, you can create partitioning rules that distribute older transactions into separate files using partitioning. Then periodically, you can archive the historical transaction data that doesn't need to be readily available to your application. For more information, go to https://dev.mysql.com/doc/refman/5.6/en/partitioning.htmlin the MySQL documentation.

To determine the file size of a table

Use the following SQL command to determine if any of your tables are too large and are candidates for partitioning.

```
SELECT TABLE_SCHEMA, TABLE_NAME,
       round(((DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024), 2) As "Approximate
size (MB)"
FROM information_schema.TABLES
WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema');
```

To enable InnoDB file-per-table tablespaces

- To enable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to `1` in the parameter group for the DB instance.

To disable InnoDB file-per-table tablespaces

- To disable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to `0` in the parameter group for the DB instance.

For information on updating a parameter group, see Working with DB Parameter Groups (p. 523).

When you have enabled or disabled InnoDB file-per-table tablespaces, you can issue an `ALTER TABLE` command to move a table from the global tablespace to its own tablespace, or from its own tablespace to the global tablespace as shown in the following example:

```
ALTER TABLE table_name ENGINE=InnoDB;
```

Creating a DB Instance Running the MySQL Database Engine

The basic building block of Amazon RDS is the DB instance. The DB instance is where you create your MySQL databases.

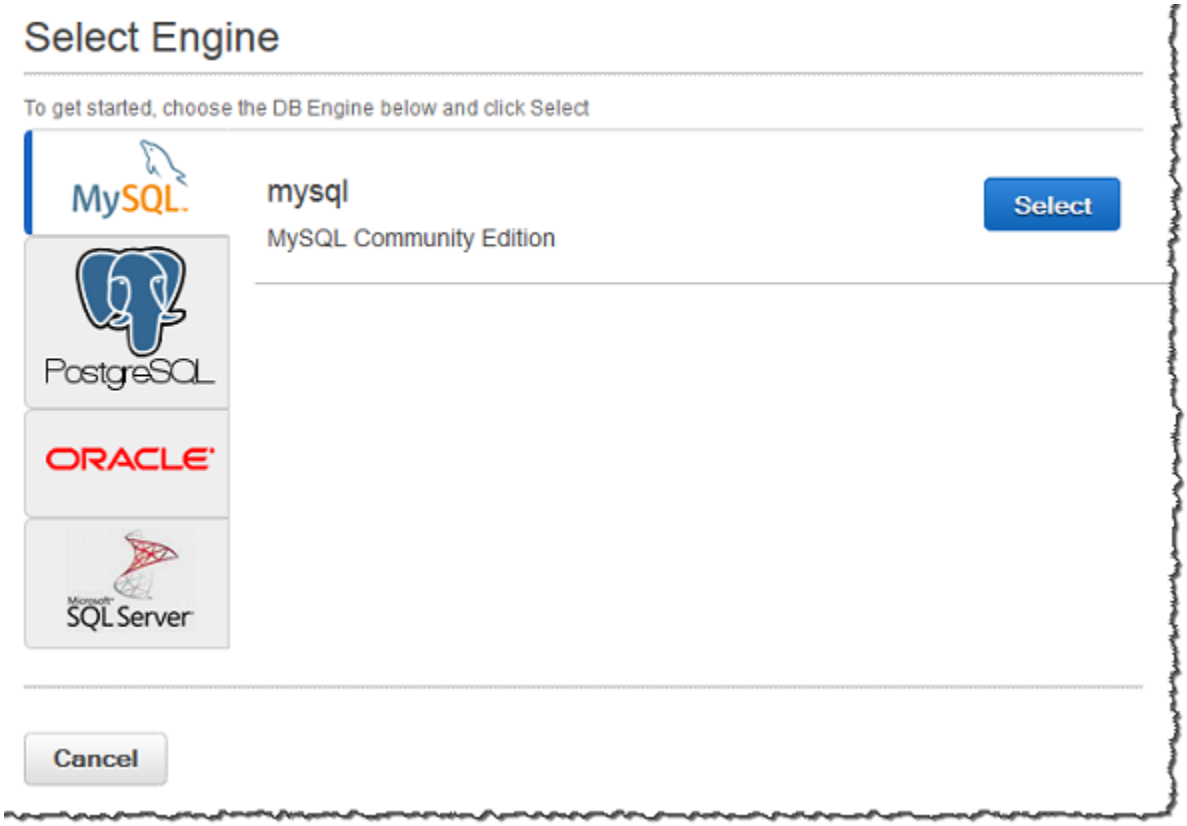
Important
You must complete the tasks in the Setting Up for Amazon RDS (p. 7) section before you can create or connect to a DB instance.

AWS Management Console

To launch a MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the top right corner of the AWS Management Console, select the region in which you want to create the DB instance.
3. In the navigation pane, click **DB Instances**.
4. Click **Launch DB Instance** to start the **Launch DB Instance Wizard**.

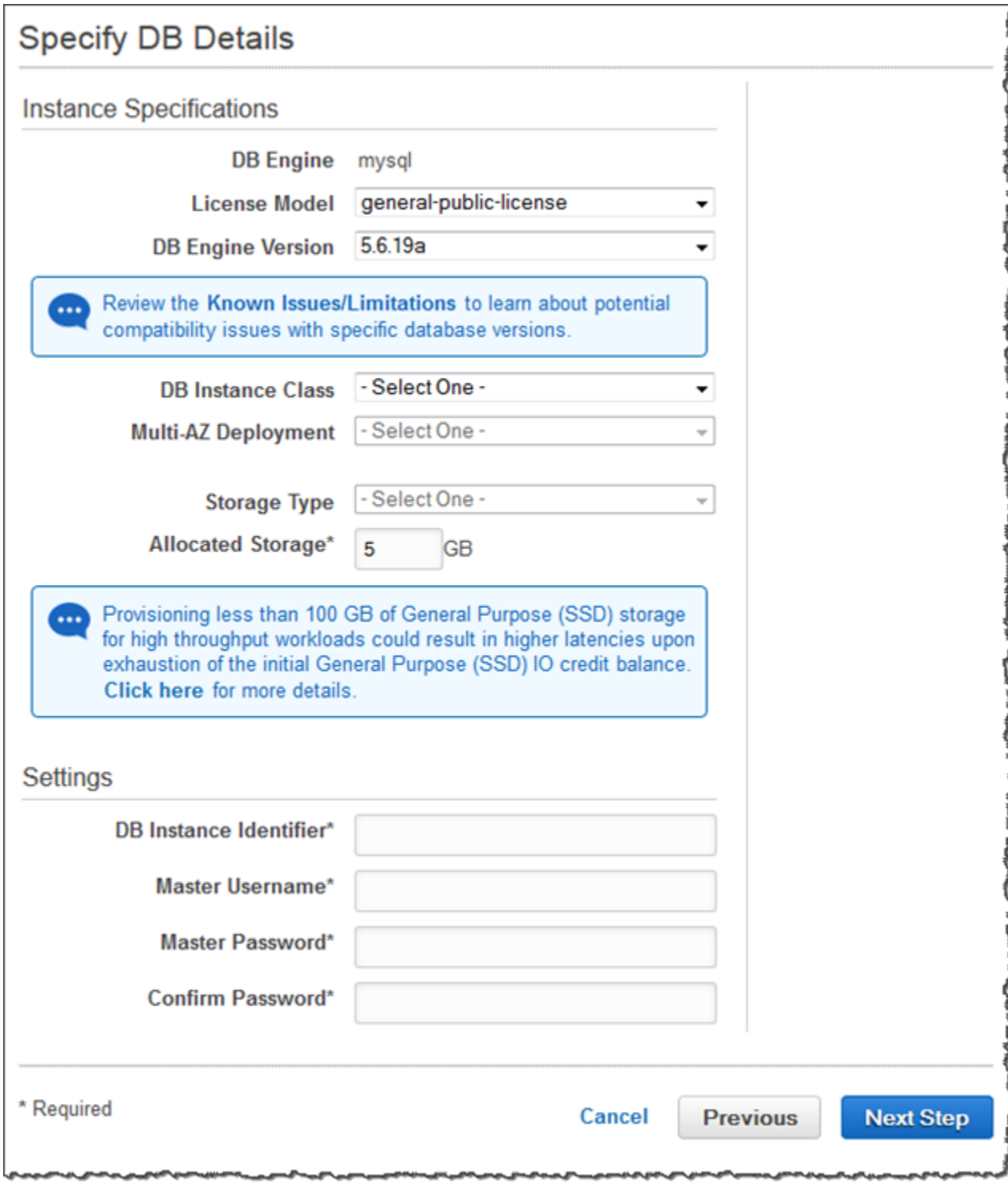
   The wizard opens on the **Select Engine** page.



5. In the **Launch DB Instance Wizard** window, click the **Select** button for the MySQL DB engine.
6. The next step asks if you are planning to use the DB instance you are creating for production. If you are, select **Yes**. By selecting **Yes**, the failover option **Multi-AZ** and the **Provisioned IOPS** storage option will be preselected in the following step. Click **Next** when you are finished.
7. On the **Specify DB Details** page, specify your DB instance information. The following table shows settings for an example DB instance. Click **Next** when you are finished.

For this parameter...	...Do this:
License Model	MySQL has only one license model. Select the default, **`General-Public-License`**, to use the general license agreement for MySQL.
DB Engine Version	Select the version of MySQL that you want to work with. Note that Amazon RDS supports several versions of MySQL.
DB Instance Class	Select a DB instance class that defines the processing and memory requirements for the DB instance. For more information about all the DB instance class options, see DB Instance Class (p. 65).
Multi-AZ Deployment	Determine if you want to create a standby replica of your DB instance in another Availability Zone for failover support. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 69).

For this parameter...	...Do this:
Allocated Storage	Type a value to allocate storage for your database (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon RDS Storage Types (p. 77).
Storage Type	Select the storage type you want to use. For more information about storage, see Storage for Amazon RDS (p. 77).
DB Instance Identifier	Type a name for the DB instance that is unique for your account in the region you selected. You may choose to add some intelligence to the name such as including the region and DB Engine you selected, for example **`mysql-instance1`**.
Master Username	Type a name using alphanumeric characters that you will use as the master user name to log on to your DB instance. The default privileges granted to the master user name account include: create, drop, references, event, alter, delete, index, insert, select, update, create temporary tables, lock tables, trigger, create view, show view, alter routine, create routine, execute, create user, process, show databases, grant option.
Master Password	Type a password that contains from 8 to 16 printable ASCII characters (excluding /,", and @) for your master user password.
Confirm Password	Re-type the Master Password for confirmation.



8. On the **Configure Advanced Settings** page, provide additional information that RDS needs to launch the MySQL DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then click Next Step.

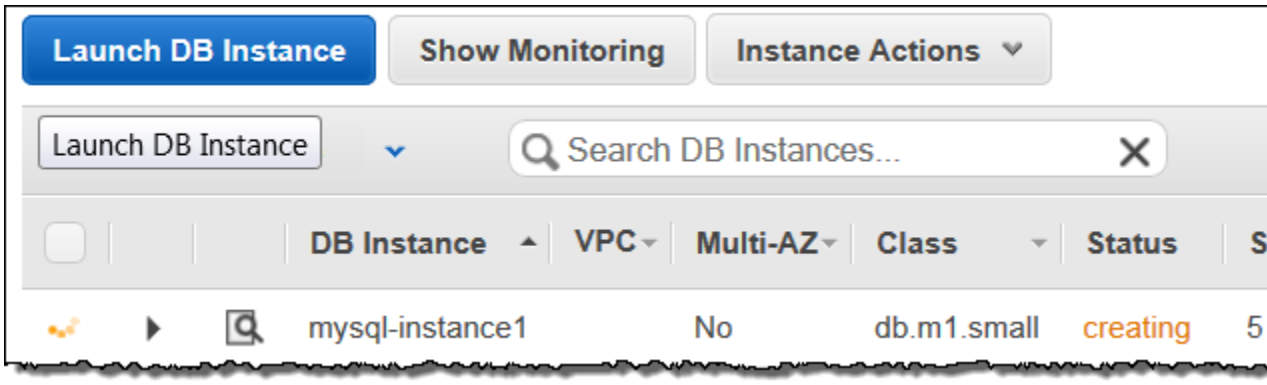
For this parameter...	...Do this:
VPC	Select the name of the Virtual Private Cloud (VPC) that will host your MySQL DB instance. If your DB instance will not be hosted in a VPC, select **Not in VPC**. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).
Availability Zone	Determine if you want to specify a particular Availability Zone. If you selected **Yes** for the Multi-AZ Deployment parameter on the previous page, you will not have any options here. For more information about Availability Zones, see Regions and Availability Zones (p. 69).
DB Security Groups	Select the security group you want to use with this DB instance. For more information about security groups, see Working with DB Security Groups (p. 537).
Database Name	Type a name for your database of 1 to 64 alpha-numeric characters. If you do not provide a name, Amazon RDS will not create a database on the DB instance you are creating.
Database Port	Specify the port that applications and utilities will use to access the database. MySQL installations default to port 3306. The firewalls at some companies block connections to the default MySQL port. If your company firewall blocks the default port, choose another port for the new DB instance.
DB Parameter Group	Select a parameter group. Each MySQL version has a default parameter group you can use, or you can create your own parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 523).
Option Group	Select an option group. Each MySQL version has a default option group you can use, or you can create your own option group. For more information about option groups, see Working with Option Groups (p. 510).
Enable Encryption	Select `Yes` to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 106).
Backup Retention Period	Select the number of days for Amazon RDS to automatically back up your DB instance. You can recover your database to any point in time during that retention period. For more information, see DB Instance Backups (p. 73).
Backup Window	Specify the period of time during which your DB instance is backed up. During the backup window, storage I/O may be suspended while your data is being backed up and you may experience elevated latency. This I/O suspension typically lasts for the duration of the snapshot. This period of I/O suspension is shorter for Multi-AZ DB deployments, since the backup is taken from the standby, but latency can occur during the backup process. For more information, see DB Instance Backups (p. 73).

For this parameter...	...Do this:
Auto Minor Version Upgrade	Select **Yes** if you want to enable your DB instance to receive minor DB Engine version upgrades automatically when they become available.
Maintenance Window	Select the weekly time range during which system maintenance can occur. For more information about the maintenance window, see Adjusting the Preferred Maintenance Window (p. 453).

In addition, Federated Storage Engine is currently not supported by Amazon RDS for MySQL.

Note
The Point-In-Time-Restore and Snapshot Restore features of Amazon RDS for MySQL require a crash recoverable storage engine, and these two features are supported only for the InnoDB storage engine. While MySQL supports multiple storage engines with varying capabilities, not all of them are optimized for crash recovery and data durability. For example, the MyISAM storage engine does not support reliable crash recovery and may result in lost or corrupt data when MySQL is restarted after a crash, preventing Point-In-Time-Restore or Snapshot restore from working as intended.
If you would like to convert existing MyISAM tables to InnoDB tables, you can use the alter table command (e.g., alter table TABLE_NAME engine=innodb;). Note that MyISAM and InnoDB have different strengths and weaknesses, so you should fully evaluate the impact of making this switch on your applications before doing so.

9. Click **Launch DB Instance** to create your MySQL DB instance.
10. On the final page of the wizard, click **Close**.
11. On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new instance to be available.



CLI

To create a MySQL DB instance

- Use the CLI command `rds-create-db-instance` to create a DB instance. For more information, go to rds-create-db-instance in the Amazon Relational Database Service Command Line Reference. For example:

```
PROMPT>rds-create-db-instance mydbinstance -s 20 -c db.m1.small -e MySQL
   - u <masterawsuser> -p <masteruserpassword> --backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE  mydbinstance  db.m1.small  mysql  20  sa  creating  3  ****  n
  5.1.57
SECGROUP  default  active
```

```
PARAMGRP  default.mysql5.1  in-sync
```

API

To create a MySQL DB instance

- Call the API action `CreateDBInstance` to create a DB instance. For more information, go to CreateDBInstance in the Amazon Relational Database Service API Reference. For example:
  - *`DBInstanceIdentifier`* = `mydbinstance`
  - *`DBInstanceClass`* = `db.m1.small`
  - *`AllocatedStorage`* = `20`
  - *`BackupRetentionPeriod`* = `3`
  - *`MasterUsername`* = `<masterawsuser>`
  - *`MasterUserPassword`* = `<masteruserpassword>`

  Example

```
https://rds.us-west-2.amazonaws.com/
    ?Action=CreateDBInstance
    &AllocatedStorage=20
    &BackupRetentionPeriod=3
    &DBInstanceClass=db.m1.small
    &DBInstanceIdentifier=mydbinstance
    &DBName=mydatabase
    &DBSecurityGroups.member.1=mysecuritygroup
    &DBSubnetGroup=mydbsubnetgroup
    &Engine=mysql
    &MasterUserPassword=<masteruserpassword>
    &MasterUsername=<masterawsuser>
    &Version=2013-09-09
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140213/us-west-2/rds/aws4_request
    &X-Amz-Date=20140213T162136Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
    &X-Amz-Signa
ture=8052a76dfb18469393c5f0182cdab0ebc224a9c7c5c949155376c1c250fc7ec3
```

Related Topics

- Amazon RDS DB Instances (p. 64)
- DB Instance Class (p. 65)
- Deleting a DB Instance (p. 459)

Connecting to a DB Instance Running the MySQL Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard MySQL client application or utility to connect to the instance. In the connection string, you specify the DNS address from the DB instance endpoint as the host parameter, and specify the port number from the DB instance endpoint as the port parameter.

You can use the AWS Management Console, the rds-describe-db-instances CLI command, or the DescribeDBInstances API action to list the details of an Amazon RDS DB instance, including its endpoint. If an endpoint value is `myinstance.123456789012.us-east-1.rds.amazonaws.com:3306`, then you would specify the following values in a MySQL connection string:

- For host or host name, specify `myinstance.123456789012.us-east-1.rds.amazonaws.com`
- For port, specify `3306`

You can connect to an Amazon RDS MySQL DB instance by using tools like the MySQL command line utility. For more information on using the MySQL utility, go to mysql - The MySQL Command Line Tool in the MySQL documentation. One GUI-based application you can use to connect is MySQL Workbench. For more information, go to the Download MySQL Workbench page.

Two common causes of connection failures to a new DB instance are:

- The DB instance was created using a security group that does not authorize connections from the device or Amazon EC2 instance where the MySQL application or utility is running. If the DB instance was created in a VPC, it must have a VPC security group that authorizes the connections. If the DB instance was created outside of a VPC, it must have a DB security group that authorizes the connections.
- The DB instance was created using the default port of 3306, and your company has firewall rules blocking connections to that port from devices in your company network. To fix this failure, recreate the instance with a different port.

You can use SSL encryption on connections to an Amazon RDS MySQL DB instance. For information, see Using SSL with a MySQL DB Instance (p. 123).

Connecting from the MySQL Utility

To connect to a DB instance using the MySQL utility

- Type the following command at a command prompt to connect to a DB instance using the MySQL utility. For the -h parameter, substitute the DNS name for your DB instance. For the -P parameter, substitute the port for your DB instance. Enter the master user password when prompted.

```
PROMPT> mysql -h myinstance.123456789012.us-east-1.rds.amazonaws.com -P 3306
 -u mymasteruser -p
```

You will see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 5.1.32-log MySQL Community Server (GPL)
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Connecting with SSL

To connect to a DB instance with SSL using the MySQL utility

Amazon RDS creates an SSL certificate for your DB instance when the instance is created. If you enable SSL certificate verification, then the SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. To connect to your DB instance using SSL, follow these steps:

1. A root certificate that works for all regions can be downloaded here.
2. Type the following command at a command prompt to connect to a DB instance with SSL using the MySQL utility. For the -h parameter, substitute the DNS name for your DB instance. For the --ssl-ca parameter, substitute the SSL certificate file name as appropriate.

   ```
   PROMPT> mysql -h myinstance.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=rds-ca-2015-root.pem
   ```

3. Include the `--ssl-verify-server-cert` parameter so that the SSL connection verifies the DB instance endpoint against the endpoint in the SSL certificate. For example:

   ```
   PROMPT> mysql -h myinstance.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=rds-ca-2015-root.pem --ssl-verify-server-cert
   ```

4. Enter the master user password when prompted.

You will see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 5.1.32-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Maximum MySQL connections

The maximum number of connections allowed to an Amazon RDS MySQL DB instance is based on the amount of memory available for the DB instance class of the DB instance. A DB instance class with more memory available will result in a larger amount of connections available. For more information on DB instance classes, see DB Instance Class (p. 65).

The connection limit for a DB instance is set by default to the maximum for the DB instance class for the DB instance. You can limit the number of concurrent connections to any value up to the maximum number

of connections allowed using the `max_connections` parameter in the parameter group for the DB instance. For more information, see Working with DB Parameter Groups (p. 523).

You can retrieve the maximum number of connections allowed for an Amazon RDS MySQL DB instance by executing the following query on your DB instance:

```
SELECT @@max_connections;
```

You can retrieve the number of active connections to an Amazon RDS MySQL DB instance by executing the following query on your DB instance:

```
SHOW STATUS WHERE `variable_name` = 'Threads_connected';
```

Related Topics

- Amazon RDS DB Instances (p. 64)
- Creating a DB Instance Running the MySQL Database Engine (p. 129)
- Amazon RDS Security Groups (p. 110)
- Deleting a DB Instance (p. 459)

Modifying a DB Instance Running the MySQL Database Engine

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. This topic guides you through modifying an Amazon RDS MySQL DB instance, and describes the settings for MySQL instances. For information about additional tasks, such as renaming, rebooting, deleting, tagging, or upgrading an Amazon RDS DB instance, see Amazon RDS DB Instance Lifecycle (p. 440). We recommend that you test any changes on a test instance before modifying a production instance so you better understand the impact of a change. This is especially important when upgrading database versions.

You can have the changes apply immediately or have them applied during the DB instance's next maintenance window. Applying changes immediately can cause an outage in some cases; for more information on the impact of the **Apply Immediately** option when modifying a DB instance, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 455).

AWS Management Console

To modify a MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Instances**.
3. Select the check box for the DB instance that you want to change, click **Instance Actions** and then click **Modify**.
4. In the **Modify DB Instance** dialog box, change any of the following settings that you want:

Setting	Description
DB Engine Version	In the list provided, click the version of the MySQL database engine that you want to use.
DB Instance Class	In the list provided, click the DB instance class that you want to use. For information about instance classes, see DB Instance Class (p. 65).
Multi-AZ Deployment	If you want to deploy your DB instance in multiple Availability Zones, click **Yes**; otherwise, click **No** .
Allocated Storage	Specify how much storage, in gigabytes, to allocate for your DB instance. The minimum allowable value is 5 GB; the maximum is 6 TB. Note that you can only increase the amount of storage when modifying a DB instance, you cannot reduce the amount of storage allocated.
Storage Type	Select the storage type you want to use. Changing from **Magnetic** to **General Purpose (SSD)** or **Provisioned IOPS (SSD)** will result in an outage. Also, changing from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic** will result in an outage. For more information about storage, see Storage for Amazon RDS (p. 77).

Setting	Description
DB Instance Identifier	You can rename the DB instance by typing a new name. When you change the DB instance identifier, an instance reboot will occur immediately if you set `Apply Immediately` to true, or will occur during the next maintenance window if you set `Apply Immediately` to false. This value is stored as a lowercase string.
New Master Password	Type a password for your master user. The password must contain from 8 to 41 alphanumeric characters.
Security Group	Select the security group you want associated with the DB instance. For more information about security groups, see Working with DB Security Groups (p. 537).
Parameter Group	Select the parameter group you want associated with the DB instance. Changing this setting does not result in an outage. The parameter group name itself is changed immediately, but the actual parameter changes are not applied until you reboot the instance without failover. The DB instance will NOT be rebooted automatically and the parameter changes will NOT be applied during the next maintenance window. For more information about parameter groups, see Working with DB Parameter Groups (p. 523).
Option Group	Select the option group you want associated with the DB instance. For more information about option groups, see Working with Option Groups (p. 510).
Backup Retention Period	Specify the number of days that automatic backups will be retained. To disable automatic backups, set this value to 0. **Note** An immediate outage will occur if you change the backup retention period from 0 to a non-zero value or from a non-zero value to 0.
Backup Window	Set the time range during which automated backups of your databases will occur. Specify a start time in Universal Coordinated Time (UTC) and a duration in hours.
Auto Minor Version Upgrade	If you want your DB instance to receive minor engine version upgrades automatically when they become available, click **Yes**. Upgrades are installed only during your scheduled maintenance window.
Maintenance Window	Set the time range during which system maintenance, including upgrades, will occur. Specify a start time in UTC and a duration in hours.

5. To apply the changes immediately, select the **Apply Immediately** check box. Selecting this option can cause an outage in some cases; for more information on the impact of the **Apply Immediately** option, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 455).
6. When all the changes are as you want them, click **Continue**. If instead you want to cancel any changes that you didn't apply in the previous step, click **Cancel**.

CLI

To modify a MySQL DB instance

- Use the command `rds-modify-db-instance`.

API

To modify a MySQL DB instance

- Use the `ModifyDBInstance` `action`.

Importing and Exporting Data From a MySQL DB Instance

We recommend using the procedures in this section to import data into or export it from a MySQL DB instance. You can use these procedures to import data from other MySQL DB instances, MySQL instances running external to Amazon RDS, and other types of data sources. To use replication to export data to an instance of MySQL that is running external to Amazon RDS, we recommend using the procedure discussed in Using Replication to Export MySQL 5.6 Data (p. 164)

Overview

We recommend the following procedures for importing data into a MySQL DB instance in the situations described:

- To import data from an existing database in a MySQL DB instance, you can create a Read Replica, and then promote the Read Replica. For more information, see Working with PostgreSQL and MySQL Read Replicas (p. 472).
- To move small amounts of MySQL data, or where service interruption on the source MySQL database isn't an issue, you can use a simple procedure to copy the data directly to your Amazon RDS MySQL DB instance using a command-line utility. For more information, see Importing Data from a MySQL DB to an Amazon RDS MySQL DB Instance (p. 146).
- To move large amounts of MySQL data, or when you want to minimize service interruption for live sites or applications that use an external MySQL instance, you can back up the data, copy it to Amazon Elastic Compute Cloud (Amazon EC2), and import it into an Amazon RDS MySQL DB instance. You can then use replication to bring the two instances into sync for any data that has been added to the source system since the copy to Amazon EC2. For more information Importing Data to an Amazon RDS MySQL DB Instance with Reduced Downtime (p. 147).
- For data in sources other than an existing MySQL database, you can create flat files and import them using the `mysqlimport` utility. For more information, see Importing Data From Any Source to a MySQL DB Instance (p. 159).
- To set up replication using an existing MySQL DB instance as the replication master, see Replication with a MySQL Instance Running External to Amazon RDS (p. 162).

> **Note**
> The 'mysql' system database contains authentication and authorization information required to log into your DB instance and access your data. Dropping, altering, renaming, or truncating tables, data, or other contents of the 'mysql' database in your DB instance can result in error and may render the DB instance and your data inaccessible. If this occurs, the DB instance can be restored from a snapshot using rds-restore-db-instance-from-db-snapshot or recovered using rds-restore-db-instance-to-point-in-time.

Importing Data Considerations

This section contains additional technical information related to loading data into MySQL. It is intended for advanced users who are familiar with the MySQL server architecture. Note that all comments related to LOAD DATA LOCAL INFILE apply to `mysqlimport` as well.

Binary Log

Data loads incur a performance penalty and require additional free disk space (up to 4X more) when binary logging is enabled versus loading the same data with binary logging turned off. The severity of the

performance penalty and the amount of free disk space required is directly proportional to the size of the transactions used to load the data.

Transaction Size

Transaction size plays an important role in MySQL data loads. It has a major influence on resource consumption, disk space utilization, resume process, time to recover, and input format (flat files or SQL). This section describes how transaction size affects binary logging and makes the case for disabling binary logging during large data loads. As noted earlier, binary logging is enabled and disabled by setting the Amazon RDS automated backup retention period. Non-zero values enable binary logging, and zero disables it. We also describe the impact of large transactions on InnoDB and why it's important to keep transaction sizes small.

Small Transactions

For small transactions, binary logging doubles the number of disk writes required to load the data. Depending upon the upload rate, other database activity taking place during the load, and the capacity of your Amazon RDS DB instance, this can severely degrade performance for other database sessions and increase the time required to load the data.

The binary logs also consume disk space roughly equal to the amount of data loaded until they are backed up and removed. Fortunately, Amazon RDS minimizes this by backing up and removing binary logs on a frequent basis.

Large Transactions

Large transactions incur a 3X penalty for IOPS and disk consumption with binary logging enabled. This is due to the binary log cache spilling to disk, consuming disk space and incurring additional IO for each write. The cache cannot be written to the binlog until the transaction commits or rolls back, so it consumes disk space in proportion to the amount of data loaded. When the transaction commits, the cache must be copied to the binlog, creating a third copy of the data on disk.

Because of this, there must be at least three times as much free disk space available to load the data compared to loading with binary logging disabled. For example, 10GB of data loaded as a single transaction will consume at least 30GB disk space during the load: 10GB for the table + 10GB for the binary log cache + 10GB for the binary log itself. The cache file remains on disk until the session that created it terminates or the session fills its binary log cache again during another transaction. The binary log must remain on disk until backed up, so it may be some time before the extra 20GB is freed.

If the data was loaded using LOAD DATA LOCAL INFILE, yet another copy of the data is created if the database has to be recovered from a backup made prior to the load. During recovery, MySQL extracts the data from the binary log into a flat file and then executes LOAD DATA LOCAL INFILE, just as the original transaction, only this time the input file is local to the database server. Continuing with the example above, recovery will fail unless there is at least 40GB free disk space available.

Disable Binary Logging

Whenever possible, disable binary logging during large data loads to avoid the resource overhead and addition disk space requirements. In Amazon RDS, disabling binary logging is as simple as setting the backup retention period to zero. If you do this, it's recommended that you take a DB Snapshot of the database instance immediately before the load so that you can quickly and easily undo changes made during loading if the need arises.

After the load, set the backup retention period back to an appropriate (no zero) value.

You cannot set the backup retention period to zero if the DB instance is a source DB instance for Read Replicas.

InnoDB

The information in this section provides a strong argument for keeping transaction sizes small when using InnoDB.

Undo

InnoDB generates undo to support features such as transaction rollback and MVCC . Undo is stored in the InnoDB system tablespace (usually ibdata1) and is retained until removed by the purge thread. The purge thread cannot advance beyond the undo of the oldest active transaction, so it is effectively blocked until the transaction commits or completes a rollback. If the database is processing other transactions during the load, their undo also accumulates in the system tablespace and cannot be removed even if they commit and no other transaction needs the undo for MVCC. In this situation, all transactions (including read-only transactions) that access any of the rows changed by any transaction (not just the load transaction) slow down as they scan through undo that could have been purged if not for the long running load transaction.

Since undo is stored in the system tablespace and since the system tablespace never shrinks in size, large data load transactions can cause the system tablespace to become quite large, consuming disk space that cannot be reclaimed without recreating the database from scratch.

Rollback

InnoDB is optimized for commits. Rolling back a large transaction can take a very, very long time. In some cases, it may be faster to perform a point-in-time recovery or restore a DB Snapshot.

Input Data Format

MySQL can accept incoming data in one of two forms: flat files and SQL. This section points out some key advantages and disadvantages of each.

Flat Files

Loading flat files with LOAD DATA LOCAL INFILE can be the fastest and least costly method of loading data as long as transactions are kept relatively small. Compared to loading the same data with SQL, flat files usually require less network traffic, lowering transmission costs and load much faster due to the reduced overhead in the database.

One Big Transaction

LOAD DATA LOCAL INFILE loads the entire flat file as one transaction. This isn't necessarily a bad thing. If the size of the individual files can be kept small, this has a number of advantages:

- Resume Capability - Keeping track of which files have been loaded is easy. If a problem arises during the load, you can pick up where you left off with little effort. Some data may have to be retransmitted to Amazon RDS, but with small files, the amount retransmitted is minimal.
- Load data in parallel - If you've got IOPs and network bandwidth to spare with a single file load, loading in parallel may save time.
- Throttle the load rate - Data load impacting other processes? Throttle the load by increasing the interval between files.

Be Careful

The advantages of LOAD DATA LOCAL INFILE diminish rapidly as transaction size increases. If breaking up a large set of data into smaller ones isn't an option, SQL may be the better choice.

SQL

SQL has one main advantage over flat files: it's easy to keep transaction sizes small. However, SQL can take significantly longer to load than flat files and it can be difficult to determine where to resume the load after a failure. For example, mysqldump files are not restartable. If a failure occurs while loading a mysqldump file, the file will require modification or replacement before the load can resume. The alternative is to restore to the point in time prior to the load and replay the file once the cause of the failure has been corrected.

Take Checkpoints Using Amazon RDS Snapshots

If you have a load that's going to take several hours or even days, loading without binary logging isn't a very attractive prospect unless you can take periodic checkpoints. This is where the Amazon RDS DB Snapshot feature comes in very handy. A DB Snapshot creates a point-in-time consistent copy of your database instance which can be used restore the database to that point in time after a crash or other mishap.

To create a checkpoint, simply take a DB Snapshot. Any previous DB Snapshots taken for checkpoints can be removed without affecting durability or restore time.

Snapshots are fast too, so frequent checkpointing doesn't add significantly to load time.

Decreasing Load Time

Here are some additional tips to reduce load times:

- Create all secondary indexes prior to loading. This is counter-intuitive for those familiar with other databases. Adding or modifying a secondary index causes MySQL to create a new table with the index changes, copy the data from the existing table to the new table, and drop the original table.
- Load data in PK order. This is particularly helpful for InnoDB tables where load times can be reduced by 75-80% and data file size cut in half.
- Disable foreign key constraints foreign_key_checks=0 For flat files loaded with LOAD DATA LOCAL INFILE, this is required in many cases. For any load, disabling FK checks will provide significant performance gains. Just be sure to enable the constraints and verify the data after the load.
- Load in parallel unless already near a resource limit. Use partitioned tables when appropriate.
- Use multi-value inserts when loading with SQL to minimize statement execution overhead. When using mysqldump, this is done automatically.
- Reduce InnoDB log IO innodb_flush_log_at_trx_commit=0

> **Note**
> Using innodb_flush_log_at_trx_commit=0 causes InnoDB to flush its logs every second instead of at each commit. This provides a significant speed advantage, but can lead to data loss during a crash. Use with caution.

Importing Data from a MySQL DB to an Amazon RDS MySQL DB Instance

The simplest way to import data from an existing MySQL database to an Amazon RDS MySQL DB instance is to copy the database with mysqldump and pipe it directly into the Amazon RDS MySQL instance. The `mysqldump` command-line utility is commonly used to make backups and transfer data from one MySQL server to another. It is included with MySQL client software.

The following example copies the `world` sample database on the local host to an Amazon RDS MySQL DB instance.

```
sudo mysqldump –u <local_user> --databases world --single-transaction --compress
 --order-by-primary  -p<local_password> |
          mysql –u <RDS_user_name> –-port=3306 --host=hostname –p<RDS_password>
```

Note
Make sure there is not a space between the `-p` option and the entered password.

Use the `–-host`, `–-user (-u)`, `--port` and `–p` options in the `mysql` command to specify the hostname, username, port, and password to connect to your Amazon RDS MySQL DB instance. The host name is the DNS name from the Amazon RDS MySQL DB instance endpoint, for example, `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.

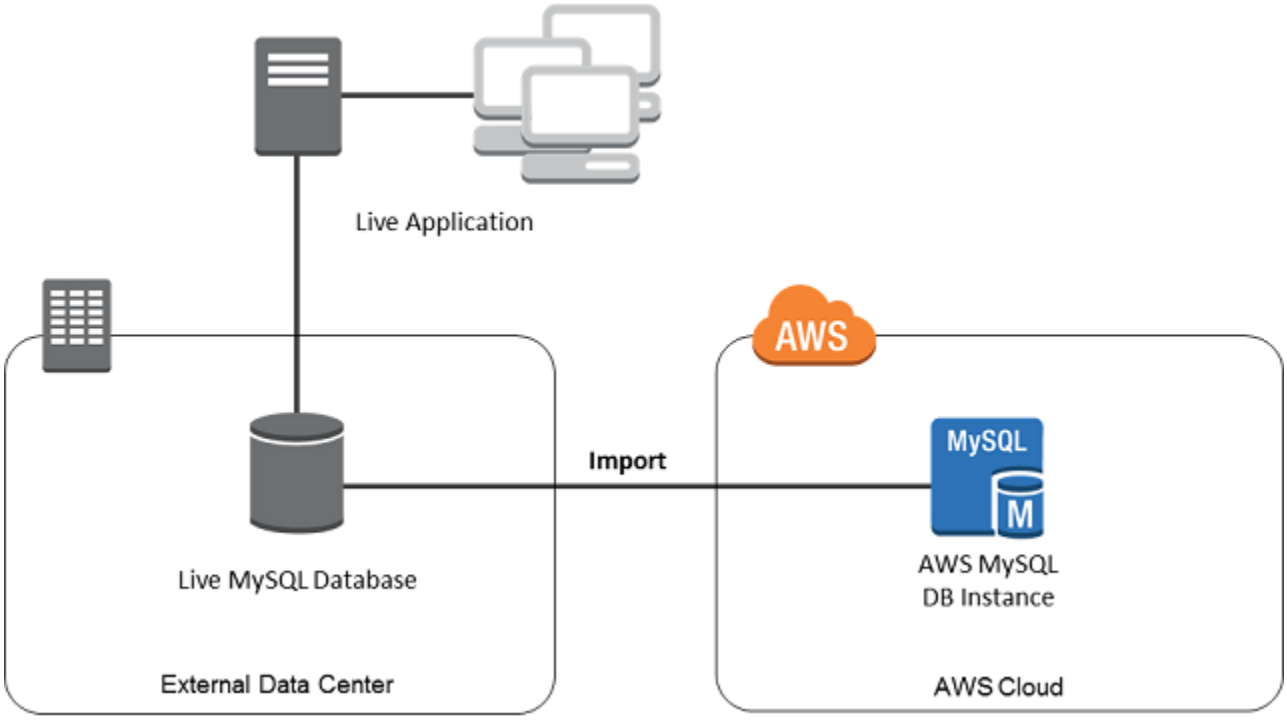
The additional `mysqldump` options that were specified to help improve operation performance and data integrity work as follows:

- Sort each table's data by its primary key using the `--order-by-primary` parameter. Taking this approach can dramatically reduce load times.
- Compress the data before sending it to Amazon RDS using the `--compress` parameter. This option can reduce network bandwidth consumption.
- Ensure that all of the data is consistent with a single point in time using the `--single-transaction` parameter. If there are other processes changing the data while `mysqldump` is reading it, use this option to maintain data integrity.
- You must create any stored procedures, triggers, functions, or events manually in your Amazon RDS database. If you have any of these objects in the database that you are copying, then exclude them when you run `mysqldump` by including the following arguments with your `mysqldump` command: `--routines=0 --triggers=0 --events=0`.

Importing Data to an Amazon RDS MySQL DB Instance with Reduced Downtime

When importing data from a MySQL database that supports a live application to an Amazon RDS MySQL DB instance, you can use the following procedure to minimize the impact on application availability. This procedure can also help if you are working with a very large database, because you can reduce the cost of the import by reducing the amount of data that is passed across the network to AWS.

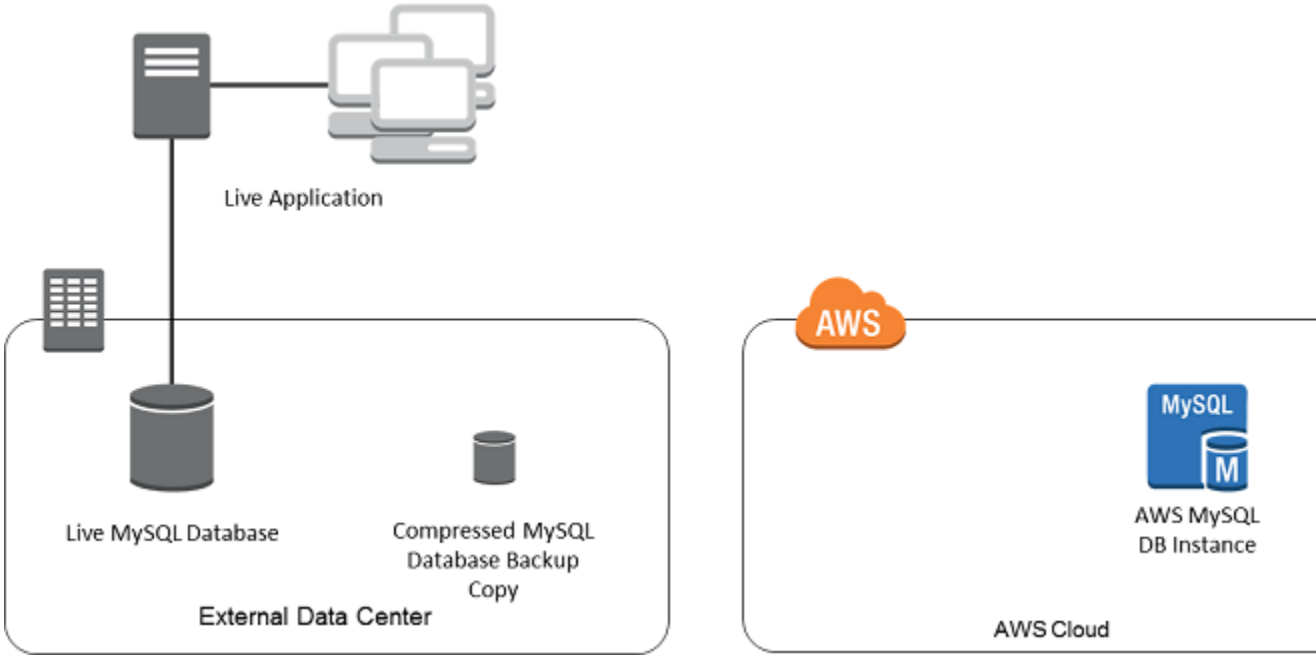
In this procedure, you will transfer a copy of your database data to an Amazon EC2 instance and import the data into a new Amazon RDS MySQL DB instance. You will then use replication to bring the Amazon RDS MySQL DB instance up-to-date with your live MySQL database, before directing your application to the Amazon RDS MySQL DB instance.



Note
We don't recommend that you use this procedure with source MySQL databases from MySQL versions earlier than version 5.1, due to potential replication issues. For more information, go to Replication Compatibility Between MySQL Versions in the MySQL documentation.

Create a Copy of Your Existing Database

The first step in the process of migrating a large amount of data to an Amazon RDS MySQL DB instance with minimal downtime is to create a copy of the source data.



You can use the `mysqldump` utility to create a database backup in either SQL or delimited-text format. You should do a test run with each format in a non-production environment to see which method minimizes the amount of time that `mysqldump` runs.

You should also weigh `mysqldump` performance against the benefit offered by using the delimited-text format for loading. A backup using delimited-text format creates a tab-separated text file for each table being dumped. You can load these files in parallel using the `LOAD DATA LOCAL INFILE` command to reduce the amount of time required to import your database. For more information about choosing a `mysqldump` format and then loading the data, go to Using mysqldump For Backups in the MySQL documentation.

Before you start the backup operation, you must set the replication options on the MySQL database that you are copying to Amazon RDS. The replication options include enabling binary logging and setting a unique server ID. Setting these options will cause your server to start logging database transactions and prepare it to be a replication master later in this process.

Note
Your database needs to be stopped to set the replication options and be in read-only mode while the backup copy is created, so you will need to schedule a maintenance window for these operations.

To set replication options

1. From a command shell, stop the `mysql` service:

```
sudo service mysqld stop
```

2. Edit the my.cnf file (this file is usually under `/etc`):

```
sudo vi /etc/my.cnf
```

Add the `log_bin` and `server_id` options to the `[mysqld]` section. The `log_bin` option provides a file name identifier for binary log files. The `server_id` option provides a unique identifier for the server in master-replica relationships.

The following example shows the updated `[mysqld]` section of a my.cnf file:

```
[mysqld]
log-bin=mysql-bin
server-id=1
```

For more information, go to Setting the Replication Master Configuration in the MySQL documentation.

3. Start the `mysql` service:

```
sudo service mysqld start
```

To create a backup copy of your existing database

1. Create a backup of your data using the `mysqldump` utility, specifying either SQL or delimited-text format.

   You must specify `--master-data=2` in order to create a backup file that can be used to start replication between servers. For more information, go to the mysqldump documentation.

   To improve performance and ensure data integrity, use the `--order-by-primary` and `--single-transaction` options of `mysqldump`.

   To avoid including the MySQL system database in the backup, do not use the `--all-databases` option with `mysqldump`. For more information, go to Creating a Dump Snapshot Using mysqldump in the MySQL documentation.

   Use `chmod` if necessary to make sure that the directory where the backup file is being created is writeable.

   - To produce SQL output, use the following command:

     ```
     sudo mysqldump --databases <database_name> --master-data=2  --single-trans
     action
     --order-by-primary -r backup.sql –u <local_user> -p
     ```

   - To produce delimited-text output, use the following command:

     ```
     sudo mysqldump --tab=<target_directory> --fields-terminated-by=, --fields-
     enclosed-by='"'
     --lines-terminated-by=0x0d0a <database_name> --master-data=2 --single-
     transaction
     --order-by-primary -p
     ```

     Note
     You must create any stored procedures, triggers, functions, or events manually in your Amazon RDS database. If you have any of these objects in the database that you are copying, exclude them when you run `mysqldump` by including the following arguments with your `mysqldump` command: `--routines=0 --triggers=0 --events=0`.

When using the delimited-text format, a CHANGE MASTER TO comment will be returned when you run `mysqldump`. This comment contains the master log file name and position. Note the values for MASTER_LOG_FILE and MASTER_LOG_POS; you will need these values when setting up replication.

```
-- Position to start replication or point-in-time recovery from
--
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000001', MAS
TER_LOG_POS=107;
```

2. Compress the copied data to reduce the amount of network resources needed to copy your MySQL data to an Amazon RDS MySQL DB. Take note of the size of the backup file, you will need this information when determining how large an Amazon EC2 instance to create. When you are done, compress the backup file using GZIP or your preferred compression utility.
   - To compress SQL output, use the following command:

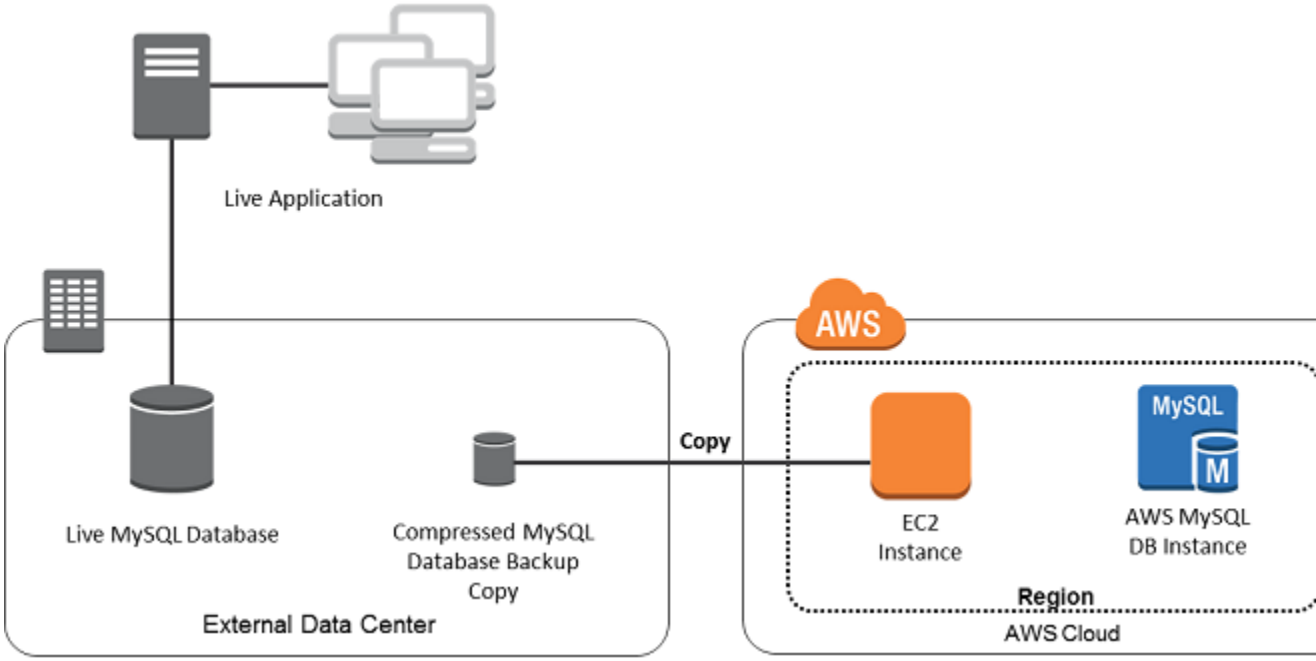
     ```
     gzip backup.sql
     ```

   - To compress delimited-text output, use the following command:

     ```
     tar -zcvf backup.tar.gz <target_directory>
     ```

Create an Amazon EC2 Instance and Copy the Compressed Database

Copying your compressed database backup file to an Amazon EC2 instance takes fewer network resources than doing a direct copy of uncompressed data between MySQL instances. Once your data is in Amazon EC2, you can copy it from there directly to your Amazon RDS MySQL DB instance. Note that, in order for you to save on the cost of network resources, your Amazon EC2 instance must be in the same region as your Amazon RDS MySQL DB instance. Having the Amazon EC2 instance in the same region as your Amazon RDS MySQL DB instance also lets you reduce network latency during the import.



To create an Amazon EC2 instance and copy your data

1. In the region where you will create your Amazon RDS instance, create an Amazon Virtual Private Cloud (VPC), a VPC security group, and a VPC subnet. Ensure that the inbound rules for your VPC security group allow the IP addresses required for your application to connect to AWS. This can be a range of IP addresses (for example `203.0.113.0/24`), or another VPC security group. You can use the Amazon VPC Console to create and manage VPCs, subnets, and security groups. For more information, go to Getting Started with Amazon VPC in the *Amazon Virtual Private Cloud Getting Started Guide*.

   Note
   Older AWS accounts can also launch instances in Amazon EC2-Classic mode. In this case, make sure that the inbound rules in the DB security group for your Amazon RDS instance allow access for your EC2-Classic instance using the Amazon EC2 private IP address. For more information, see Working with DB Security Groups (p. 537).

2. Open the Amazon EC2 Console and select the region that will contain both your Amazon EC2 instance and your Amazon RDS MySQL DB instance. Launch an Amazon EC2 instance using the VPC, subnet, and security group that you created in Step 1. Ensure that you select an instance type with enough storage for your database backup file when it is uncompressed. For details on Amazon EC2 instances, go to Getting Started with Amazon EC2 Linux Instances in the *Amazon Elastic Compute Cloud User Guide for Linux*.
3. Edit the VPC security group and add the private IP address for your new Amazon EC2 instance. The private IP address will be used when connecting to your Amazon RDS MySQL DB instance. You can find the private IP address on the **Details** tab of the **Instance** pane in the Amazon EC2 Console. For more information on modifying a VPC security group, go to Security Groups for Your VPC in the *Amazon Virtual Private Cloud User Guide*.
4. Copy your compressed database backup file from your local system to your Amazon EC2 instance. Use `chmod` if necessary to make sure you have write permission for the target directory of the Amazon EC2 instance. You can use `scp` or an SSH client to copy the file. The following is an example:

```
$ scp -r -i <key pair>.pem backup.sql.gz ec2-user@<EC2 DNS>:/<target_direct
ory>/backup.sql.gz
```

Important
Be sure to copy sensitive data using a secure network transfer protocol.

5. Connect to your Amazon EC2 instance and install the latest updates and the MySQL client tools using the following commands:

```
sudo yum update –y
sudo yum install mysql-server –y
```

   For more information, go to Connect to Your Instance in the *Amazon Elastic Compute Cloud User Guide for Linux*.

6. While connected to your Amazon EC2 instance, decompress your database backup file. For example:
   - To decompress SQL output, use the following command:

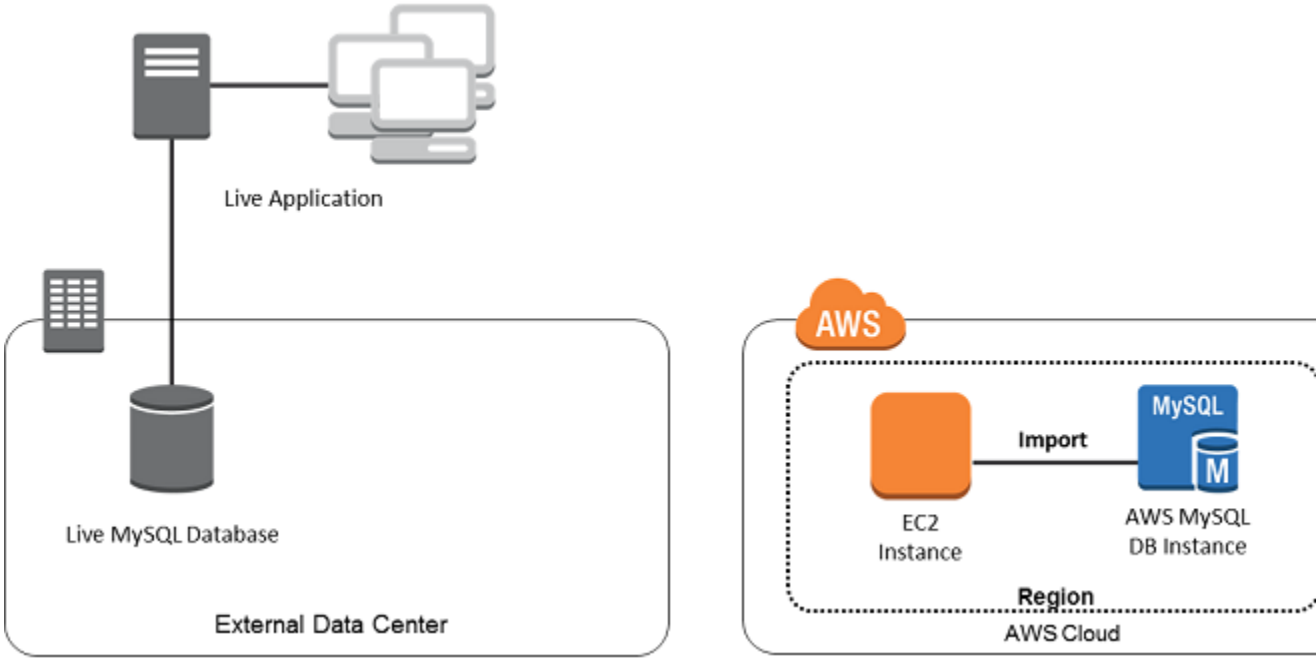
```
gzip backup.sql.gz –d
```

   - To decompress delimited-text output, use the following command:

```
tar xzvf backup.tar.gz
```

Create an Amazon RDS MySQL DB instance and Import Data from Your Amazon EC2 Instance

By creating an Amazon RDS MySQL DB instance in the same region as your Amazon EC2 instance, you can import the database backup file from Amazon EC2 faster than you can import it over the Internet.



To create an Amazon RDS MySQL DB instance and import your data

1. Determine which DB instance class and what amount of storage space is required to support the expected workload for this Amazon RDS MySQL DB instance. This process should include deciding what is sufficient space and processing capacity for your data load procedures, and also what is required to handle the production workload. You can estimate this based on the size and resources of the source MySQL database. For more information, see DB Instance Class (p. 65).
2. Determine if Amazon RDS provisioned input/output operations per second (IOPS) is required to support the workloads. Provisioned IOPS storage delivers fast throughput for online transaction processing (OLTP) workloads, which are I/O intensive. For more information, see Amazon RDS Provisioned IOPS Storage to Improve Performance (p. 82).
3. Open the Amazon RDS Console. In the upper-right corner, select the region that contains your Amazon EC2 instance.
4. Click **Launch a DB Instance**, and then go through the steps to select options for your DB instance:
   a. On the **Select Engine** page, click **MySQL**.
   b. On the **Do you plan to use this database for production purposes?** page, click **No** to skip configuring Multi-AZ deployment and provisioned IOPS storage.
   c. In the **Instance Specifications** section of the **Specify DB Details** page, specify the DB instance class and allocated storage size that you have determined are appropriate. Select **No** for Multi-AZ Deployment. Specify whether or not to use Provisioned IOPS as you determined in Step 2. For **DB Engine Version**, select the version that is compatible with your source MySQL instance, as follows:
      - If your source MySQL instance is 5.1.x, the Amazon RDS MySQL DB instance must be 5.5.x.
      - If your source MySQL instance is 5.5.x, the Amazon RDS MySQL DB instance can be 5.5.x or 5.6.23 or later.
      - If your source MySQL instance is 5.6.x, the Amazon RDS MySQL DB instance must be 5.6.x.

        Important
        If your source MySQL instance runs a version prior to version 5.6.4, or if the source MySQL instance was upgraded from a version prior to version 5.6.4, then you must create an Amazon RDS MySQL DB instance running version 5.6.23 or later.

      Accept the default values for all other boxes in this section.

      In the **Settings** section, specify the requested database and user information. Click **Next** when you are done.
   d. In the **Network & Security** section of the **Configure Advanced Settings** page, select the same VPC and VPC security group as for your Amazon EC2 instance. This approach will ensure that your Amazon EC2 instance and your Amazon RDS instance are visible to each other over the network. Accept the default values for all other boxes in this section.

      In the **Database Options** section, specify a database name. Accept the default values for all other boxes in this section.

      In the **Backup** section, set the backup retention period to 0. Accept the default values for all other boxes in this section.

      In the **Maintenance** section, accept the default values for all of the boxes. Click **Launch Instance** when you are done.

   Do not configure multiple Availability Zones, backup retention, or Read Replicas until after you have imported the database backup. When that import is done, you can set Multi-AZ and backup retention the way you want them for the production instance. For a detailed walkthrough of creating an Amazon RDS MySQL DB instance, see Creating a DB Instance Running the MySQL Database Engine (p. 129).
5. Review the default configuration options for the Amazon RDS MySQL DB instance. In the left navigation pane of the Amazon RDS Management Console, click on **Parameter Groups** , and then click on the magnifying glass icon next to the **default.mysqlx.x** parameter group. If this parameter group does not

have the configuration options that you want, find a different one that does, or create a new parameter group. For more information on creating a parameter group, see Working with DB Parameter Groups (p. 523). If you decide to use a different parameter group than the default, associate it with your Amazon RDS MySQL DB instance. For more information, see Modifying a DB Instance Running the MySQL Database Engine (p. 140).

6. Connect to the new Amazon RDS MySQL DB instance as the master user, and create the users required to support the administrators, applications, and services that will need to access the instance. The host name for the Amazon RDS MySQL DB instance will be the **Endpoint** value for this instance without including the port number, for example, `mysampledb.claxc2oy9ak1.us-west-2.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.
7. Connect to your Amazon EC2 instance. For more information, go to Connect to Your Instance in the *Amazon Elastic Compute Cloud User Guide for Linux*.
8. Connect to your Amazon RDS MySQL DB instance as a remote host from your Amazon EC2 instance using the `mysql` command. The following is an example:

```
mysql –h <host_name> -port=3306 –u <db_master_user> -p
```

   The host name is the DNS name from the Amazon RDS MySQL DB instance endpoint.
9. At the `mysql` prompt, run the `source` command and pass it the name of your database dump file to load the data into the Amazon RDS MySQL DB instance.
   - For SQL format, use the following command:

```
mysql> source backup.sql;
```

   - For delimited-text format, first create the database (if it isn't the default database you created when setting up the Amazon RDS MySQL DB instance):

```
$ mysql> create database <database_name>;
$ mysql> use <database_name>;
```

     Then create the tables:

```
$ mysql> source <table1>.sql
$ mysql> source <table2>.sql
etc…
```

     Then import the data:

```
$ mysql> LOAD DATA LOCAL INFILE 'table1.txt' INTO TABLE table1 FIELDS TER
MINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
$ mysql> LOAD DATA LOCAL INFILE 'table2.txt' INTO TABLE table2 FIELDS TER
MINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
etc…
```

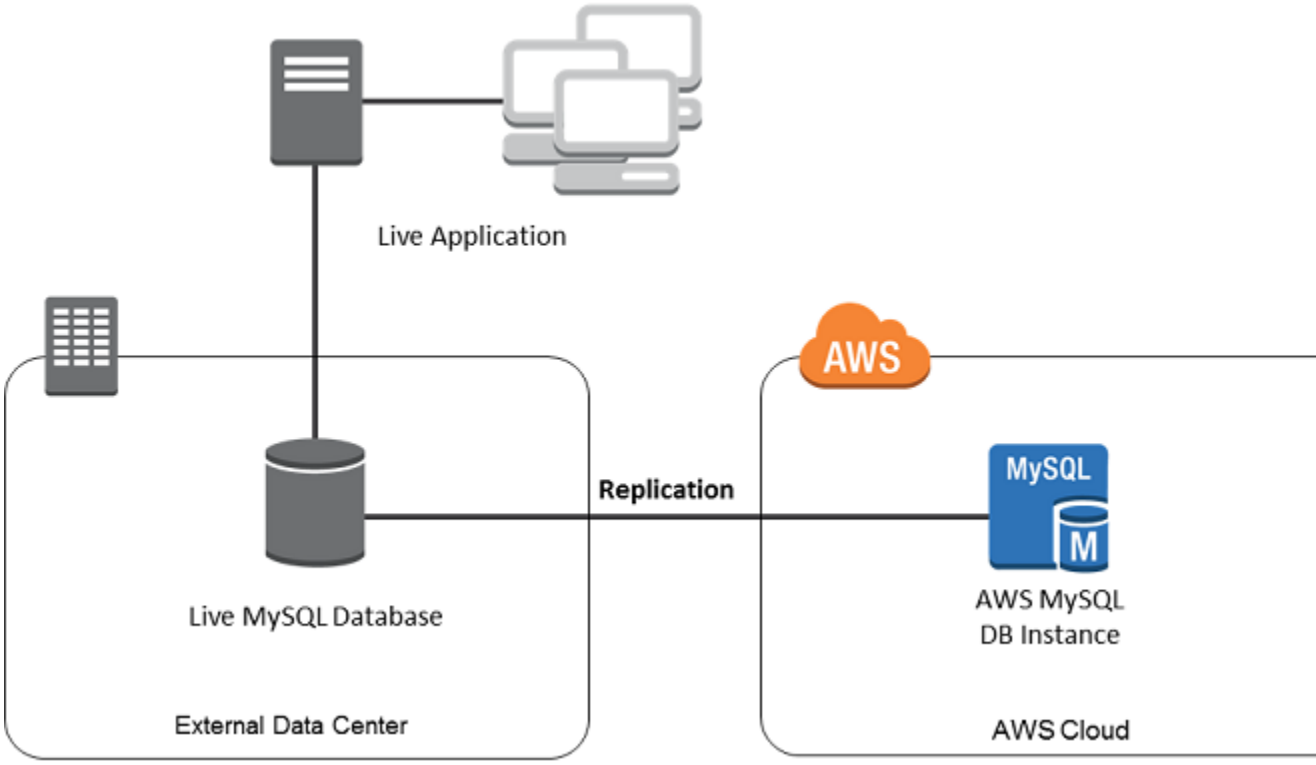
     To improve performance, you can perform these operations in parallel from multiple connections so that all of your tables get created and then loaded at the same time.

     Note
     If you used any data-formatting options with `mysqldump` when you initially dumped the table, you must use the same options with `mysqlimport` or LOAD DATA LOCAL INFILE to ensure proper interpretation of the data file contents.

10. Run a simple SELECT query against one or two of the tables in the imported database to verify that the import was successful.
11. This procedure no longer requires the Amazon EC2 instance. If you no longer need the Amazon EC2 instance that you imported your data from, then you can terminate it.

Replicate Between Your MySQL Database and New Amazon RDS MySQL DB Instance

Because your source database was likely updated during the time that it took to copy and transfer the data to the Amazon RDS MySQL DB instance, you will use replication to bring the copied database up-to-date with the source database.



Note
The permissions required to start replication on an Amazon RDS MySQL DB instance are restricted and not available to your Amazon RDS master user. Because of this, you must use the Amazon RDS mysql.rds_set_external_master (p. 178) and mysql.rds_start_replication (p. 180) commands to set up replication between your live database and your Amazon RDS MySQL database.

To start replication

Earlier, you enabled binary logging and set a unique server ID for your source MySQL database. Now you can set up your Amazon RDS MySQL DB instance as a replica with your live database as the replication master.

1. In the Amazon RDS Management Console, add the IP address of the server that hosts the source MySQL database to the VPC security group for the Amazon RDS MySQL DB instance. For more information on modifying a VPC security group, go to Security Groups for Your VPC in the *Amazon Virtual Private Cloud User Guide*.

You might also need to configure your local network to permit connections from the IP address of your Amazon RDS MySQL DB instance, so that it can communicate with your source MySQL instance. To find the IP address of the Amazon RDS MySQL DB instance, use the `host` command:

```
host <RDS_MySQL_DB_host_name>
```

The host name is the DNS name from the Amazon RDS MySQL DB instance endpoint, for example, `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.

2. Using the client of your choice, connect to the source MySQL instance and create a MySQL user that will be used for replication. This account is used solely for replication and must be restricted to your domain to improve security. The following is an example:

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>';
```

3. For the source MySQL instance, grant `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges to your replication user. For example, to grant the `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges on all databases for the '`repl_user`' user for your domain, issue the following command:

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydo
main.com' IDENTIFIED BY '<password>';
```

4. If you used SQL format to create your backup file, look at the contents of that file:

```
cat backup.sql
```

The file will include a CHANGE MASTER TO comment that contains the master log file name and position. This comment is included in the backup file when you use the `--master-date` option with `mysqldump`. Note the values for MASTER_LOG_FILE and MASTER_LOG_POS.

```
--
-- Position to start replication or point-in-time recovery from
--

-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000001', MAS
TER_LOG_POS=107;
```

5. Make the Amazon RDS MySQL DB instance the replica. Connect to the Amazon RDS MySQL DB instance as the master user and identify the source MySQL database as the replication master by using the mysql.rds_set_external_master (p. 178) command. Use the master log file name and master log position that you determined in the previous step if you have a SQL format backup file, or that you determined when creating the backup files if you used delimited-text format. The following is an example:

```
CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306,
    'repl_user', '<password>', 'mysql-bin-changelog.000001', 107, 0);
```

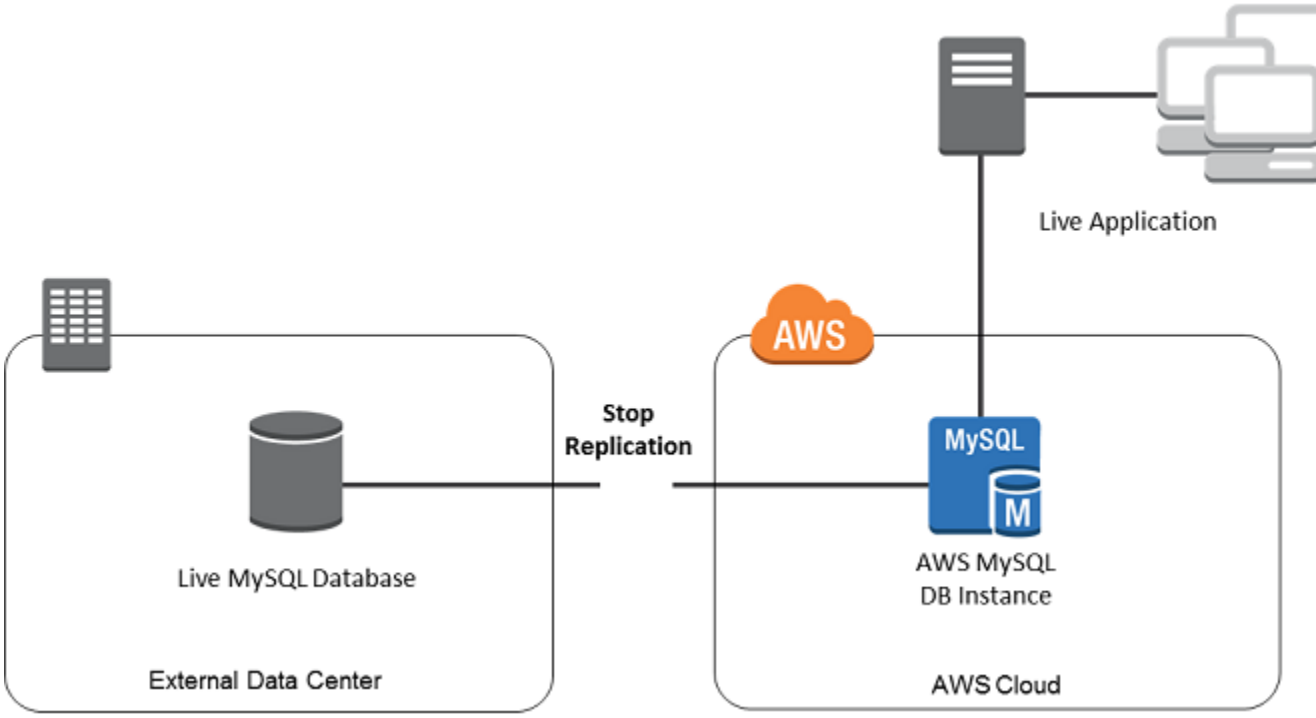
6. On the Amazon RDS MySQL DB instance, issue the mysql.rds_start_replication (p. 180) command to start replication:

```
CALL mysql.rds_start_replication;
```

7. On the Amazon RDS MySQL DB instance, run the SHOW SLAVE STATUS command to determine when the replica is up-to-date with the replication master. The results of the SHOW SLAVE STATUS command include the Seconds_Behind_Master field. When the Seconds_Behind_Master field returns 0, then the replica is up-to-date with the master.
8. After the Amazon RDS MySQL DB instance is up-to-date, enable automated backups so you can restore that database if needed. You can enable or modify automated backups for your Amazon RDS MySQL DB instance using the Amazon RDS Management Console. For more information, see Working With Automated Backups (p. 496).

Redirect Your Live Application to Your Amazon RDS MySQL Instance

Once the Amazon RDS MySQL DB instance is up-to-date with the replication master, you can now update your live application to use the Amazon RDS instance.



To redirect your live application to your Amazon RDS MySQL instance and stop replication

1. To add the VPC security group for the Amazon RDS MySQL DB instance, add the IP address of the server that hosts the application. For more information on modifying a VPC security group, go to Security Groups for Your VPC in the *Amazon Virtual Private Cloud User Guide*.
2. Verify that the Seconds_Behind_Master field in the SHOW SLAVE STATUS command results is 0, which indicates that the replica is up-to-date with the replication master:

```
SHOW SLAVE STATUS;
```

3. Stop replication for the Amazon RDS instance using the mysql.rds_stop_replication (p. 181) command:

```
CALL mysql.rds_stop_replication;
```

4. Update your application to use the Amazon RDS MySQL DB instance. This update will typically involve changing the connection settings to identify the host name and port of the Amazon RDS MySQL DB instance, the user account and password to connect with, and the database to use.
5. Run the mysql.rds_reset_external_master (p. 180) command on your Amazon RDS MySQL DB instance to reset the replication configuration so this instance is no longer identified as a replica:

```
CALL mysql.rds_reset_external_master;
```

6. Enable additional Amazon RDS MySQL features such as Multi-AZ support and Read Replicas. For more information, see High Availability (Multi-AZ) (p. 71) and Working with PostgreSQL and MySQL Read Replicas (p. 472).

Importing Data From Any Source to a MySQL DB Instance

If you have more than 1GB of data to load, or if your data is coming from somewhere other than a MySQL database, we recommend creating flat files and loading them with mysqlimport. mysqlimport is another command line utility bundled with the MySQL client software whose purpose is to load flat files into MySQL. For information about mysqlimport, go to mysqlimport - A Data Import Program in the MySQL documentation.

We also recommend creating DB Snapshots of the target Amazon RDS DB instance before and after the data load. Amazon RDS DB Snapshots are complete backups of your DB instance that can be used to restore your DB instance to a known state. When you initiate a DB Snapshot, I/O operations to your database instance are momentarily suspended while your database is backed up.

Creating a DB Snapshot immediately before the load allows you restore the database to its state prior to the load, should the need arise. A DB Snapshot taken immediately after the load protects you from having to load the data again in case of a mishap and can also be used to seed new database instances.

The following list shows the steps to take. Each step is discussed in more detail below.

1. Create flat files containing the data to be loaded.
2. Stop any applications accessing the target DB instance.
3. Create a DB Snapshot.
4. Consider disabling Amazon RDS automated backups.
5. Load the data using mysqlimport.
6. Enable automated backups again.

Step 1: Create Flat Files Containing the Data to be Loaded

Use a common format, such as CSV (Comma-Separated Values), to store the data to be loaded. Each table must have its own file; data for multiple tables cannot be combined in the same file. Give each file the same name as the table it corresponds to. The file extension can be anything you like. For example, if the table name is "sales", the file name could be "sales.csv" or "sales.txt", but not "sales_01.csv".

Whenever possible, order the data by the primary key of the table being loaded. This drastically improves load times and minimizes disk storage requirements.

The speed and efficiency of this procedure is dependent upon keeping the size of the files small. If the uncompressed size of any individual file is larger than 1GB, split it into multiple files and load each one separately.

On Unix-like systems (including Linux), use the 'split' command. For example, the following command splits the sales.csv file into multiple files of less than 1GB, splitting only at line breaks (-C 1024m). The new files will be named sales.part_00, sales.part_01, etc.

```
split -C 1024m -d sales.csv sales.part_
```

Similar utilities are available on other operating systems.

Step 2: Stop Any Applications Accessing the Target DB Instance

Before starting a large load, stop all application activity accessing the target DB instance that you will be loading to (particularly if other sessions will be modifying the tables being loaded or tables they reference). This will reduce the risk of constraint violations occurring during the load, improve load performance, and make it possible to restore the database instance to the point just prior to the load without losing changes made by processes not involved in the load.

Of course, this may not be possible or practical. If you are unable to stop applications from accessing the DB instance prior to the load, take steps to ensure the availability and integrity of your data. The specific steps required vary greatly depending upon specific use cases and site requirements.

Step 3: Create a DB Snapshot

If you will be loading data into a new DB instance that contains no data, you may skip this step. Otherwise, creating a DB Snapshot of your DB instance will allow you to restore the database Instance to the point just prior to the load, if it becomes necessary. As previously mentioned, when you initiate a DB Snapshot, I/O operations to your database instance are suspended for a few minutes while the database is backed up.

In the example below, we use the rds-create-db-snapshot command to create a DB Snapshot of our AcmeRDS instance and give the DB Snapshot the identifier "preload".

```
rds-create-db-snapshot AcmeRDS --db-snapshot-identifier=preload
```

You can also use the restore from DB Snapshot functionality in order to create test database instances for dry runs or to "undo" changes made during the load.

It is important to keep in mind that restoring a database from a DB Snapshot creates a new DB instance which, like all DB instances, has a unique identifier and endpoint. If you need to restore the database instance without changing the endpoint, you must first delete the DB instance so that the endpoint can be reused.

For example, to create a DB instance for dry runs or other testing, you would give the DB instance its own identifier. In the example, "AcmeRDS-2" is the identifier and we would connect to the database instance using the endpoint associated with AcmeRDS-2.

```
rds-restore-db-instance-from-db-snapshot AcmeRDS-2 --db-snapshot-identifier=pre
load
```

To reuse the existing endpoint, we must first delete the database instance and then give the restored database the same identifier:

```
rds-delete-db-instance AcmeRDS --final-db-snapshot-identifier AcmeRDS-Final
rds-restore-db-instance-from-db-snapshot AcmeRDS --db-snapshot-identifier=preload
```

Note that the example takes a final DB Snapshot of the database instance before deleting it. This is optional, but recommended.

Step 4: Consider Disabling Amazon RDS Automated Backups

Warning: DO NOT DISABLE AUTOMATED BACKUPS IF YOU NEED TO RETAIN THE ABILITY TO PERFORM POINT-IN-TIME RECOVERY. Disabling automated backups erases all existing backups, so point-in-time recovery will not be possible after automated backups have been disabled. Disabling automated backups is a performance optimization and is not required for data loads. Note that DB Snapshots are not affected by disabling automated backups. All existing DB Snapshots are still available for restore.

Disabling automated backups will reduce load time by about 25% and reduce the amount of storage space required during the load. If you will be loading data into a new DB instance that contains no data, disabling backups is an easy way to speed up the load and avoid using the additional storage needed for backups. However, if you will be loading into a DB instance that already contains data; you must weigh the benefits of disabling backups against the impact of losing the ability to perform point-in-time-recovery.

DB instances have automated backups enabled by default (with a one day retention period). In order to disable automated backups, you must set the backup retention period to zero. After the load, you can re-enable backups by setting the backup retention period to a non-zero value. In order to enable or disable backups, Amazon RDS must shut the DB instance down and restart it in order to turn MySQL logging on or off.

Use the rds-modify-db-instance command to set the backup retention to zero and apply the change immediately. Setting the retention period to zero requires a DB instance restart, so wait until the restart has completed before proceeding.

```
rds-modify-db-instance AcmeRDS --apply-immediately --backup-retention-period=0
```

You can check the status of your DB instance with the rds-describe-db-instances command. The example displays the status of the AcmeRDS database instance and includes the --headers option to show column headings.

```
rds-describe-db-instances AcmeRDS --headers
```

When the Status column shows that the database is available, you're ready to proceed.

Step 5: Load the Data

Use the mysqlimport utility to load the flat files into Amazon RDS. In the example we tell mysqlimport to load all of the files named "sales" with an extension starting with "part_". This is a convenient way to load all of the files created in the "split" example. Use the --compress option to minimize network traffic. The --fields-terminated-by=',' option is used for CSV files and the --local option specifies that the incoming data is located on the client. Without the --local option, MySQL will look for the data on the database host, so always specify the --local option.

```
mysqlimport --local --compress --user=username --password --host=hostname \
--fields-terminated-by=',' Acme sales.part_*
```

For very large data loads, take additional DB Snapshots periodically between loading files and note which files have been loaded. If a problem occurs, you can easily resume from the point of the last DB Snapshot, avoiding lengthy reloads.

Step 6: Enable Amazon RDS Automated Backups

Once the load is finished, re-enable Amazon RDS automated backups by setting the backup retention period back to its pre-load value. As noted earlier, Amazon RDS will restart the DB instance, so be prepared for a brief outage.

In the example, we use the rds-modify-db-instance command to enable automated backups for the AcmeRDS DB instance and set the retention period to 1 day.

```
rds-modify-db-instance AcmeRDS --apply-immediately --backup-retention-period=1
```

Replication with a MySQL Instance Running External to Amazon RDS

You can set up replication between an Amazon RDS MySQL DB instance and a MySQL instance that is external to Amazon RDS. Be sure to follow these guidelines when you set up an external replication master and a replica on Amazon RDS:

- Monitor failover events for the Amazon RDS MySQL DB instance that is your replica. If a failover occurs, then the DB instance that is your replica might be recreated on a new host with a different network address. For information on how to monitor failover events, see Using Amazon RDS Event Notification (p. 575).
- Maintain the binlogs on your master instance until you have verified that they have been applied to the replica. This maintenance ensures that you can restore your master instance in the event of a failure.
- Turn on automated backups on your MySQL DB instance on Amazon RDS. Turning on automated backups ensures that you can restore your replica to a particular point in time if you need to re-synchronize your master and replica. For information on backups and point-in-time restore, see Backing Up and Restoring (p. 495).

Note
The permissions required to start replication on an Amazon RDS MySQL DB instance are restricted and not available to your Amazon RDS master user. Because of this, you must use the Amazon RDS mysql.rds_set_external_master (p. 178) and mysql.rds_start_replication (p. 180) commands to set up replication between your live database and your Amazon RDS MySQL database.

Start replication between an external master instance and a MySQL DB instance on Amazon RDS

1. Make the source MySQL instance read-only:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Run the `SHOW MASTER STATUS` command on the source MySQL instance to determine the binlog location. You will receive output similar to the following example:

```
File                          Position
-----------------------------------
 mysql-bin-changelog.000031      107
-----------------------------------
```

3. Copy the database from the external MySQL instance to the Amazon RDS MySQL DB instance using `mysqldump`. For very large databases, you might want to use the procedure in Importing Data to an Amazon RDS MySQL DB Instance with Reduced Downtime (p. 147).

```
mysqldump --databases <database_name> --single-transaction --compress --order-
by-primary
     –u <local_user> -p<local_password> | mysql --host=hostname –-port=3306
–u <RDS_user_name> –p<RDS_password>
```

Note
Make sure there is not a space between the `-p` option and the entered password.

Use the `–-host`, `–-user (-u)`, `--port` and `–p` options in the `mysql` command to specify the hostname, username, port, and password to connect to your Amazon RDS MySQL DB instance. The host name is the DNS name from the Amazon RDS MySQL DB instance endpoint, for example, `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.

4. Make the source MySQL instance writeable again:

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

For more information on making backups for use with replication, go to Backing Up a Master or Slave by Making It Read Only in the MySQL documentation.

5. In the Amazon RDS Management Console, add the IP address of the server that hosts the external MySQL database to the VPC security group for the Amazon RDS MySQL DB instance. For more information on modifying a VPC security group, go to Security Groups for Your VPC in the *Amazon Virtual Private Cloud User Guide*.

You might also need to configure your local network to permit connections from the IP address of your Amazon RDS MySQL DB instance, so that it can communicate with your external MySQL instance. To find the IP address of the Amazon RDS MySQL DB instance, use the `host` command:

```
host <RDS_MySQL_DB_host_name>
```

The host name is the DNS name from the Amazon RDS MySQL DB instance endpoint.

6. Using the client of your choice, connect to the external MySQL instance and create a MySQL user that will be used for replication. This account is used solely for replication and must be restricted to your domain to improve security. The following is an example:

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>';
```

7. For the external MySQL instance, grant `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges to your replication user. For example, to grant the `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges on all databases for the '`repl_user`' user for your domain, issue the following command:

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydo
main.com' IDENTIFIED BY '<password>';
```

8. Make the Amazon RDS MySQL DB instance the replica. Connect to the Amazon RDS MySQL DB instance as the master user and identify the external MySQL database as the replication master by using the mysql.rds_set_external_master (p. 178) command. Use the master log file name and master log position that you determined in Step 2. The following is an example:

```
CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306,
    'repl_user', '<password>', 'mysql-bin-changelog.000031', 107, 0);
```

9. On the Amazon RDS MySQL DB instance, issue the mysql.rds_start_replication (p. 180) command to start replication:

```
CALL mysql.rds_start_replication;
```

Using Replication to Export MySQL 5.6 Data

You can use replication to export data from a MySQL 5.6 DB instance to a MySQL instance running external to Amazon RDS. The MySQL instance external to Amazon RDS can be running either on-premises in your data center, or on an Amazon EC2 instance. The MySQL DB instance must be running version 5.6.13 or later. The MySQL instance external to Amazon RDS must be running the same version as the Amazon RDS instance, or a higher version.

Replication to an instance of MySQL running external to Amazon RDS is only supported during the time it takes to export a database from a MySQL DB instance. The replication should be terminated when the data has been exported and applications can start accessing the external instance.

The following list shows the steps to take. Each step is discussed in more detail in later sections.

1. Prepare an instance of MySQL running external to Amazon RDS.
2. Configure the MySQL DB instance to be the replication source.

3. Use `mysqldump` to transfer the database from the Amazon RDS instance to the instance external to Amazon RDS.
4. Start replication to the instance running external to Amazon RDS.
5. After the export completes, stop replication.

Prepare an Instance of MySQL External to Amazon RDS

Install an instance of MySQL external to Amazon RDS.

Connect to the instance as the master user, and create the users required to support the administrators, applications, and services that access the instance.

Follow the directions in the MySQL documentation to prepare the instance of MySQL running external to Amazon RDS as a replica. For more information, go to Setting the Replication Slave Configuration.

Configure an egress rule for the external instance to operate as a Read Replica during the export. The egress rule will allow the MySQL Read Replica to connect to the MySQL DB instance during replication. Specify an egress rule that allows TCP connections to the port and IP address of the source Amazon RDS MySQL DB instance.

If the Read Replica is running in an Amazon EC2 instance in an Amazon VPC, specify the egress rules in a VPC security group. If the Read Replica is running in an Amazon EC2 instance that is not in a VPC, specify the egress rule in an Amazon EC2 security group. If the Read Replica is installed on-premises, specify the egress rule in a firewall.

If the Read Replica is running in a VPC, configure VPC ACL rules in addition to the security group egress rule. For more information about Amazon VPC network ACLs, go to Network ACLs.

- ACL ingress rule allowing TCP traffic to ports 1024-65535 from the IP address of the source MySQL DB instance.
- ACL egress rule: allowing outbound TCP traffic to the port and IP address of the source MySQL DB instance.

Prepare the Replication Source

Prepare the MySQL DB instance as the replication source.

Ensure your client computer has enough disk space available to save the binary logs while setting up replication.

Create a replication account by following the directions in Creating a User For Replication.

Configure ingress rules on the system running the replication source MySQL DB instance that will allow the external MySQL Read Replica to connect during replication. Specify an ingress rule that allows TCP connections to the port used by the Amazon RDS instance from the IP address of the MySQL Read Replica running external to Amazon RDS.

If the Amazon RDS instance is running in a VPC, specify the ingress rules in a VPC security group. If the Amazon RDS instance is not running in an in a VPC, specify the ingress rules in a database security group.

If the Amazon RDS instance is running in a VPC, configure VPC ACL rules in addition to the security group ingress rule. For more information about Amazon VPC network ACLs, go to Network ACLs.

- ACL ingress rule: allow TCP connections to the port used by the Amazon RDS instance from the IP address of the external MySQL Read Replica.

- ACL egress rule: allow TCP connections from ports 1024-65535 to the IP address of the external MySQL Read Replica.

Ensure that the backup retention period is set long enough that no binary logs are purged during the export. If any of the logs are purged before the export is complete, you must restart replication from the beginning. For more information about setting the backup retention period, see Working With Automated Backups (p. 496).

Use the `mysql.rds_set_configuration` stored procedure to set the binary log retention period long enough that the binary logs are not purged during the export. For more information, see Accessing MySQL 5.6 Binary Logs (p. 597).

To further ensure that the binary logs of the source instance are not purged, create an Amazon RDS Read Replica from the source instance. For more information, see Creating a Read Replica (p. 476). After the Amazon RDS Read Replica has been created, call the `mysql.rds_stop_replication` stored procedure to stop the replication process. The source instance will no longer purge its binary log files, so they will be available for the replication process.

Copy the Database

Run the MySQL `SHOW SLAVE STATUS` statement against the MySQL instance running external to Amazon RDS, and note the master_host, master_port, master_log_file, and read_master_log_pos values.

Use the `mysqldump` utility to create a snapshot, which copies the data from Amazon RDS to your local client computer. Then run another utility to load the data into the MySQL instance running external to RDS. Ensure your client computer has enough space to hold the `mysqldump` files from the databases to be replicated. This process can take several hours for very large databases. Follow the directions in Creating a Dump Snapshot Using mysqldump.

The following example shows how to run `mysqldump` on a client, and then pipe the dump into the `mysql` client utility, which loads the data into the external MySQL instance.

```
mysqldump -h RDS instance endpoint -u user -p password --port=3306 --single-
transaction --routines --triggers --databases  database database2 --compress
--compact | mysql -h MySQL host -u master user -p password --port 3306
```

The following example shows how to run `mysqldump` on a client and write the dump to a file.

```
mysqldump -h RDS instance endpoint -u user -p password --port=3306 --single-
transaction --routines --triggers --databases  database database2 > path/rds-
dump.sql
```

Complete the Export

After you have loaded the `mysqldump` files to create the databases on the MySQL instance running external to Amazon RDS, start replication from the source MySQL DB instance to export all source changes that have occurred after you stopped replication from the Amazon RDS Read Replica.

Use the MySQL `CHANGE MASTER` statement to configure the external MySQL instance. Specify the ID and password of the user granted REPLICATION SLAVE permissions. Specify the master_host, master_port, master_log_file, and read_master_log_pos values you got from the Mysql `SHOW SLAVE STATUS` statement you ran on the RDS Read Replica. For more information, go to Setting the Master Configuration on the Slave.

Use the MySQL `START SLAVE` command to initiate replication from the source MySQL DB instance and the MySQL replica.

Run the MySQL `SHOW SLAVE STATUS` command on the Amazon RDS instance to verify that it is operating as a Read Replica. For more information about interpreting the results, go to SHOW SLAVE STATUS Syntax.

After replication on the MySQL instance has caught up with the Amazon RDS source, use the MySQL `STOP SLAVE` command to terminate replication from the source MySQL DB instance.

On the Amazon RDS Read Replica, call the `mysql.rds_start_replication` stored procedure. This will allow Amazon RDS to start purging the binary log files from the source MySQL DB instance.

Appendix: Common DBA Tasks for MySQL

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances running the MySQL database engine. In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.

For information about working with MySQL log files on Amazon RDS, see MySQL Database Log Files (p. 594)

Topics

- Killing a Session or Query (p. 168)
- Skipping the Current Replication Error (p. 168)
- Working with InnoDB Tablespaces to Improve Crash Recovery Times (p. 169)
- Managing the Global Status History (p. 171)

Killing a Session or Query

To terminate user sessions or queries on DB instances, Amazon RDS provides the following commands:

```
PROMPT> CALL mysql.rds_kill(thread-ID)
PROMPT> CALL mysql.rds_kill_query(thread-ID)
```

For example, to kill the session that is running on thread 99, you would type the following:

```
PROMPT> CALL mysql.rds_kill(99);
```

To kill the query that is running on thread 99, you would type the following:

```
PROMPT> CALL mysql.rds_kill_query(99);
```

Skipping the Current Replication Error

Amazon RDS provides a mechanism for you to skip an error on your Read Replicas if the error is causing your Read Replica to hang and the error doesn't affect the integrity of your data.

Note
You should first verify that the error can be safely skipped. In a MySQL utility, connect to the Read Replica and run the following MySQL command:

```
SHOW SLAVE STATUS\G
```

For information about the values returned, go to SHOW SLAVE STATUS Syntax in the MySQL documentation.

To skip the error, you can issue the following command:

```
CALL mysql.rds_skip_repl_error;
```

This command has no effect if you run it on the source DB instance, or on a Read Replica that has not encountered a replication error.

For more information, such as the versions of MySQL that support `mysql.rds_skip_repl_error`, see mysql_rds_skip_repl_error (p. 182).

Important
If you attempt to call *mysql.rds_skip_repl_error* and encounter the following error: `ERROR 1305 (42000): PROCEDURE mysql.rds_skip_repl_error does not exist`, then upgrade your MySQL DB instance to the latest minor version or one of the minimum minor versions listed in mysql_rds_skip_repl_error (p. 182).

Working with InnoDB Tablespaces to Improve Crash Recovery Times

Every table in MySQL consists of a table definition, data, and indexes. The MySQL storage engine InnoDB stores table data and indexes in a *tablespace*. InnoDB creates a global shared tablespace that contains a data dictionary and other relevant metadata, and it can contain table data and indexes. InnoDB can also create separate tablespaces for each table and partition. These separate tablespaces are stored in files with a .ibd extension and the header of each tablespace contains a number that uniquely identifies it.

Amazon RDS provides a parameter in a MySQL parameter group called `innodb_file_per_table`. This parameters controls whether InnoDB adds new table data and indexes to the shared tablespace (by setting the parameter value to 0) or to individual tablespaces (by setting the parameter value to 1). Amazon RDS sets the default value for `innodb_file_per_table` parameter to 1, which allows you to drop individual InnoDB tables and reclaim storage used by those tables for the DB instance. In most use cases, setting the `innodb_file_per_table` parameter to 1 is the recommended setting.

You should set the `innodb_file_per_table` parameter to 0 when you have a large number of tables, such as over 1000 tables when you use standard (magnetic) or general purpose SSD storage or over 10,000 tables when you use Provisioned IOPS storage. When you set this parameter to 0, individual tablespaces are not created and this can improve the time it takes for database crash recovery.

MySQL processes each metadata file, which includes tablespaces, during the crash recovery cycle. The time it takes MySQL to process the metadata information in the shared tablespace is negligible compared to the time it takes to process thousands of tablespace files when there are multiple tablespaces. Because the tablespace number is stored within the header of each file, the aggregate time to read all the tablespace files can take up to several hours. For example, a million InnoDB tablespaces on standard storage can take from five to eight hours to process during a crash recovery cycle. In some cases, InnoDB can determine that it needs additional cleanup after a crash recovery cycle so it will begin another crash recovery cycle, which will extend the recovery time. Keep in mind that a crash recovery cycle also entails rolling-back transactions, fixing broken pages, and other operations in addition to the processing of tablespace information.

Since the `innodb_file_per_table` parameter resides in a parameter group, you can change the parameter value by editing the parameter group used by your DB instance without having to reboot the DB instance. After the setting is changed, for example, from 1 (create individual tables) to 0 (use shared

tablespace), new InnoDB tables will be added to the shared tablespace while existing tables continue to have individual tablespaces. To move an InnoDB table to the shared tablespace, you must use the `ALTER TABLE` command.

Migrating Multiple Tablespaces to the Shared Tablespace

You can move an InnoDB table's metadata from its own tablespace to the shared tablespace by using the following command, which will rebuild the table metadata according to the `innodb_file_per_table` parameter setting.

```
PROMPT>ALTER TABLE name ENGINE = InnoDB
```

For example, the following query returns an `ALTER TABLE` statement for every InnoDB table.

```
SELECT CONCAT('ALTER TABLE `',
REPLACE(TABLE_SCHEMA, '`', '``'), '`.`',
REPLACE(TABLE_NAME, '`', '``'), '` ENGINE=InnoDB;')
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE'
AND ENGINE = 'InnoDB' AND TABLE_SCHEMA <> 'mysql';
```

Rebuilding a MySQL table to move the table's metadata to the shared tablespace requires additional storage space temporarily to rebuild the table, so the DB instance must have storage space available. During rebuilding, the table is locked and inaccessible to queries. For small tables or tables not frequently accessed, this may not be an issue; for large tables or tables frequently accessed in a heavily concurrent environment, you can rebuild tables on a Read Replica.

You can create a Read Replica and migrate table metadata to the shared tablespace on the Read Replica. While the ALTER TABLE statement blocks access on the Read Replica, the source DB instance is not affected. The source DB instance will continue to generate its binary logs while the Read Replica lags during the table rebuilding process. Because the rebuilding requires additional storage space and the replay log file can become large, you should create a Read Replica with storage allocated that is larger than the source DB instance.

The following steps should be followed to create a Read Replica and rebuild InnoDB tables to use the shared tablespace:

1. Ensure that backup retention is enabled on the source DB instance so that binary logging is enabled
2. Use the AWS Console or RDS CLI to create a Read Replica for the source DB instance. Since the creation of a Read Replica involves many of the same processes as crash recovery, the creation process may take some time if there are a large number of InnoDB tablespaces. Allocate more storage space on the Read Replica than is currently used on the source DB instance.
3. When the Read Replica has been created, create a parameter group with the parameter settings `read_only = 0` and `innodb_file_per_table = 0`, and then associate the parameter group with the Read Replica.
4. Issue ALTER TABLE <name> ENGINE = InnoDB against all tables you want migrated on the replica.
5. When all of your ALTER TABLE statements have completed on the Read Replica, verify that the Read Replica is connected to the source DB instance and that the two instances are in-sync.
6. When ready, use the AWS Console or RDS CLI to promote the Read Replica to be the master instance. Make sure that the parameter group used for the new master has the innodb_file_per_table parameter set to 0. Change the name of the new master, and point any applications to the new master instance.

Managing the Global Status History

MySQL maintains many status variables that provide information about its operation. Their value can help you detect locking or memory issues on a DB instance . The values of these status variables are cumulative since last time the DB instance was started. You can reset most status variables to 0 by using the `FLUSH STATUS` command.

To allow for monitoring of these values over time, Amazon RDS provides a set of procedures that will snapshot the values of these status variables over time and write them to a table, along with any changes since the last snapshot. This infrastructure, called Global Status History (GoSH), is installed on all MySQL DB instances starting with versions 5.1.62 and 5.5.23. GoSH is disabled by default.

To enable GoSH, you first enable the event scheduler from a DB parameter group by setting the parameter event_scheduler to ON. For information about creating and modifying a DB parameter group, see Working with DB Parameter Groups (p. 523).

You can then use the procedures in the following table to enable and configure GoSH. For each procedure, at the command prompt, type the following:

```
PROMPT> CALL procedure-name;
```

Where *procedure-name* is one of the procedures in the table.

Procedure	Description
`rds_enable_gsh_collector`	Enables GoSH to take default snapshots at intervals specified by `rds_set_gsh_collector`.
`rds_set_gsh_collector`	Specifies the interval, in minutes, between snapshots. Default value is 5.
`rds_disable_gsh_collector`	Disables snapshots.
`rds_collect_global_status_history`	Takes a snapshot on demand.
`rds_enable_gsh_rotation`	Enables rotation of the contents of the `mysql.global_status_history` table to `mysql.global_status_history_old` at intervals specified by `rds_set_gsh_rotation`.
`rds_set_gsh_rotation`	Specifies the interval, in days, between table rotations. Default value is 7.
`rds_disable_gsh_rotation`	Disables table rotation.
`rds_rotate_global_status_history`	Rotates the contents of the `mysql.global_status_history` table to `mysql.global_status_history_old` on demand.

When GoSH is running, you can query the tables that it writes to. For example, to query the hit ratio of the Innodb buffer pool, you would issue the following query:

```
select a.collection_end, a.collection_start, (( a.variable_Delta-
```

```
b.variable_delta)/a.variable_delta)*100 as "HitRatio"
            from rds_global_status_history as a join rds_global_status_history
 as b on a.collection_end = b.collection_end
            where a. variable_name = 'Innodb_buffer_pool_read_requests' and
b.variable_name = 'Innodb_buffer_pool_reads'
```

Appendix: Options for MySQL Database Engine

This appendix describes options, or additional features, that are available for Amazon RDS instances running the MySQL DB engine. To enable these options, you can add them to a custom option group, and then associate the option group with your DB instance. For more information about working with options, see Option Groups Overview (p. 510).

The following option is currently supported for MySQL 5.6:

- MEMCACHED

MySQL 5.6 memcached Support

Amazon RDS supports using the `memcached` interface to InnoDB tables that was introduced in MySQL 5.6. The `memcached` API enables applications to use InnoDB tables in a manner similar to NoSQL key-value data stores.

> **Important**
> We recommend that you only use the `memcached` interface with MySQL version 5.6.21b or later. This is because there are a number of bug fixes related to the `memcached` interface which are included in the MySQL engine starting with version 5.6.21b. For more information, go to Changes in MySQL 5.6.20 (2014-07-31) and Changes in MySQL 5.6.21 (2014-09-23) in the MySQL documentation.

`memcached` is a simple, key-based cache. Applications use `memcached` to insert, manipulate, and retrieve key-value data pairs from the cache. MySQL 5.6 introduces a plugin that implements a daemon service that exposes data from InnoDB tables through the `memcached` protocol. For more information about the MySQL `memcached` plugin, go to InnoDB Integration with memcached.

You enable `memcached` support for an Amazon RDS MySQL 5.6 instance by:

1. Determining the security group to use for controlling access to the `memcached` interface. If the set of applications already using the SQL interface are the same set that will access the `memcached` interface, you can use the existing VPC or DB security group used by the SQL interface. If a different set of applications will access the `memcached` interface, define a new VPC or DB security group. For more information about managing security groups, see Amazon RDS Security Groups (p. 110)
2. Creating a custom DB option group, selecting MySQL as the engine type and a 5.6 version. For more information about creating an option group, see Creating an Option Group (p. 514).
3. Adding the *`MEMCACHED`* option to the option group. Specify the port that the `memcached` interface will use, and the security group to use in controlling access to the interface. For more information about adding options, see Adding an Option to an Option Group (p. 515).
4. Modifying the option settings to configure the `memcached` parameters, if necessary. For more information about how to modify option settings, see Modifying an Option Setting (p. 519).
5. Applying the option group to an instance. Amazon RDS enables `memcached` support for that instance when the option group is applied:
   - You enable `memcached` support for a new instance by specifying the custom option group when you launch the instance. For more information about launching a MySQL instance, see Creating a DB Instance Running the MySQL Database Engine (p. 129).
   - You enable `memcached` support for an existing instance by specifying the custom option group when you modify the instance. For more information about modifying a MySQL instance, see Modifying a DB Instance Running the MySQL Database Engine (p. 140).
6. Specifying which columns in your MySQL tables can be accessed through the `memcached` interface. The `memcached` plug-in creates a catalog table named `containers` in a dedicated database named `innodb_memcache`. You insert a row into the `containers` table to map an InnoDB table for access

through memcached. You specify a column in the InnoDB table that is used to store the memcached key values, and one or more columns that are used to store the data values associated with the key. You also specify a name that a memcached application uses to refer to that set of columns. For details on inserting rows in the containers table, go to Internals of the InnoDB memcached Plugin. For an example of mapping an InnoDB table and accessing it through memcached, go to Specifying the Table and Column Mappings for an InnoDB + memcached Application.

7. If the applications accessing the memcached interface are on different computers or EC2 instances than the applications using the SQL interface, add the connection information for those computers to the VPC or DB security group associated with the MySQL instance. For more information about managing security groups, see Amazon RDS Security Groups (p. 110).

You turn off the memcached support for an instance by modifying the instance and specifying the MySQL 5.6 default option group. For more information about modifying a MySQL instance, see Modifying a DB Instance Running the MySQL Database Engine (p. 140).

MySQL memcached Security Considerations

The memcached protocol does not support user authentication. For more information about MySQL memcached security considerations, go to memcached Deployment and Using memcached as a MySQL Caching Layer.

You can take the following actions to help increase the security of the memcached interface:

- Specify a different port than the default of 11211 when adding the *MEMCACHED* option to the option group.
- Ensure that you associate the memcached interface with either a VPC or DB security group that limits access to known, trusted client addresses or EC2 instances. For more information about managing security groups, see Amazon RDS Security Groups (p. 110).

MySQL memcached Connection Information

To access the memcached interface, an application must specify both the DNS name of the Amazon RDS instance and the memcached port number. For example, if an instance has a DNS name of my-cache-instance.cg034hpkmmjt.region.rds.amazonaws.com and the memcached interface is using port 11212, the connection information specified in PHP would be:

```
<?php

$cache = new Memcache;
$cache->connect('my-cache-instance.cg034hpkmmjt.region.rds.amazonaws.com',11212);
?>
```

To find the DNS name and memcached port of an Amazon RDS MySQL instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the top right corner of the AWS Management Console, select the region that contains the DB instance.
3. In the navigation pane, click **Instances**.
4. Select the arrow to the left of name of the DB Instance running the MySQL database engine. In the description display, note the value of the **endpoint** field. The DNS name is the part of the endpoint

up to the semicolon (:). Ignore the semicolon and the port number after the semicolon, that port is not used to access the `memcached` interface.

5. Note the name listed in the **Option Group(s)** field.
6. In the navigation pane, click **Option Groups**.
7. Select the arrow to the left of the name of the option group used by the MySQL DB instance. In the description display, note the value of the **port** setting in the **MEMCACHED** option.

MySQL memcached Option Settings

Amazon RDS exposes the MySQL `memcached` parameters as option settings in the Amazon RDS *`MEMCACHED`* option.

MySQL memcached Parameters

- *`DAEMON_MEMCACHED_R_BATCH_SIZE`* - an integer that specifies how many `memcached` read operations (get) to perform before doing a COMMIT to start a new transaction. The allowed values are 1 to 4294967295, the default is 1. The option does not take effect until the instance is restarted.
- *`DAEMON_MEMCACHED_W_BATCH_SIZE`* - an integer that specifies how many `memcached` write operations, such as add, set, or incr, to perform before doing a COMMIT to start a new transaction. The allowed values are 1 to 4294967295, the default is 1. The option does not take effect until the instance is restarted.
- *`INNODB_API_BK_COMMIT_INTERVAL`* - an integer that specifies how often to auto-commit idle connections that use the InnoDB `memcached` interface. The allowed values are 1 to 1073741824, the default is 5. The option takes effect immediately, without requiring that you restart the instance.
- *`INNODB_API_DISABLE_ROWLOCK`* - a Boolean that disables (1 (true)) or enables (0 (false)) the use of row locks when using the InnoDB `memcached` interface. The default is 0 (false). The option does not take effect until the instance is restarted.
- *`INNODB_API_ENABLE_MDL`* - a Boolean that when set to 0 (false) locks the table used by the InnoDB `memcached` plugin, so that it cannot be dropped or altered by DDL through the SQL interface. The default is 0 (false). The option does not take effect until the instance is restarted.
- *`INNODB_API_TRX_LEVEL`* - an integer that specifies the transaction isolation level for queries processed by the `memcached` interface. The allowed values are 0 to 3. The default is 0. The option does not take effect until the instance is restarted.

Amazon RDS configures these MySQL `memcached` parameters, they cannot be modified: *`DAEMON_MEMCACHED_LIB_NAME`*, *`DAEMON_MEMCACHED_LIB_PATH`*, and *`INNODB_API_ENABLE_BINLOG`*. The parameters that MySQL administrators set by using *`daemon_memcached_options`* are available as individual *`MEMCACHED`* option settings in Amazon RDS.

MySQL *`daemon_memcached_options`* Parameters

- *`BINDING_PROTOCOL`* - a string that specifies the binding protocol to use. The allowed values are `auto`, `ascii`, or `binary`. The default is `auto`, which means the server automatically negotiates the protocol with the client. The option does not take effect until the instance is restarted.
- *`BACKLOG_QUEUE_LIMIT`* - an integer that specifies how many network connections can be waiting to be processed by `memcached`. Increasing this limit may reduce errors received by a client that is not able to connect to the `memcached` instance, but does not improve the performance of the server. The allowed values are 1 to 2048, the default is 1024. The option does not take effect until the instance is restarted.
- *`CAS_DISABLED`* - a Boolean that enables (1 (true)) or disables (0 (false)) the use of compare and swap (CAS), which reduces the per-item size by 8 bytes. The default is 0 (false). The option does not take effect until the instance is restarted.

- *CHUNK_SIZE* - an integer that specifies the minimum chunk size, in bytes, to allocate for the smallest item's key, value, and flags. The allowed values are 1 to 48. The default is 48 and you can significantly improve memory efficiency with a lower value. The option does not take effect until the instance is restarted.
- *CHUNCK_SIZE_GROWTH_FACTOR* - a float that controls the size of new chunks. The size of a new chunk is the size of the previous chunk times CHUNCK_SIZE_GROWTH_FACTOR. The allowed values are 1 to 2, the default is 1.25. The option does not take effect until the instance is restarted.
- *ERROR_ON_MEMORY_EXHAUSTED* - a Boolean, when set to 1 (true) it specifies that memcached will return an error rather than evicting items when there is no more memory to store items. If set to 0 (false), memcached will evict items if there is no more memory. The default is 0 (false). The option does not take effect until the instance is restarted.
- *MAX_SIMULTANEOUS_CONNECTIONS* - an integer that specifies the maximum number of concurrent connections. Setting this value to anything under 10 prevents MySQL from starting. The allowed values are 10 to 1024, the default is 1024. The option does not take effect until the instance is restarted.
- *VERBOSITY* - an string that specifies the level of information logged in the MySQL error log by the memcached service. The default is v. The option does not take effect until the instance is restarted. The allowed values are:
  - *v* - Logs errors and warnings while executing the main event loop.
  - *vv* - In addition to the information logged by v, also logs each client command and the response.
  - *vvv* - In addition to the information logged by vv, also logs internal state transitions.

Amazon RDS configures these MySQL *DAEMON_MEMCAHCED_OPTIONS* parameters, they cannot be modified: *DAEMON_PROCESS*, *LARGE_MEMORY_PAGES*, *MAXIMUM_CORE_FILE_LIMIT*, *MAX_ITEM_SIZE*, *LOCK_DOWN_PAGE_MEMORY*, *MASK*, *IDFILE*, *REQUESTS_PER_EVENT*, *SOCKET*, and *USER*.

Appendix: MySQL on Amazon RDS SQL Reference

This appendix describes system stored procedures that are available for Amazon RDS instances running the MySQL DB engine.

Overview

The following system stored procedures are supported for Amazon RDS DB instances running MySQL.

Replication

- mysql.rds_set_external_master (p. 178)
- mysql.rds_reset_external_master (p. 180)
- mysql.rds_start_replication (p. 180)
- mysql.rds_stop_replication (p. 181)
- mysql_rds_skip_repl_error (p. 182)
- mysql.rds_next_master_log (p. 182)

InnoDB cache warming

- mysql.rds_innodb_buffer_pool_dump_now (p. 184)
- mysql.rds_innodb_buffer_pool_load_now (p. 185)
- mysql.rds_innodb_buffer_pool_load_abort (p. 185)

Managing additional configuration (for example, binlog file retention)

- mysql.rds_set_configuration (p. 186)
- mysql.rds_show_configuration (p. 186)

Terminating a session or query

- mysql.rds_kill (p. 187)
- mysql.rds_kill_query (p. 188)

Logging

- mysql.rds_rotate_general_log (p. 188)
- mysql.rds_rotate_slow_log (p. 189)

Managing the global status history

- mysql.rds_enable_gsh_collector (p. 190)
- mysql.rds_set_gsh_collector (p. 190)
- mysql.rds_disable_gsh_collector (p. 191)
- mysql.rds_collect_global_status_history (p. 191)
- mysql.rds_enable_gsh_rotation (p. 191)
- mysql.rds_set_gsh_rotation (p. 192)

- mysql.rds_disable_gsh_rotation (p. 192)
- mysql.rds_rotate_global_status_history (p. 193)

SQL Reference Conventions

This section explains the conventions that are used to describe the syntax of the system stored procedures and tables described in the SQL reference section.

Character	Description
UPPERCASE	Words in uppercase are keywords.
[ ]	Square brackets indicate optional arguments.
{ }	Braces indicate that you are required to choose one of the arguments inside the braces.
|	Pipes separate arguments that you can choose.
italics	Words in italics indicate placeholders. You must insert the appropriate value in place of the word in italics.
…	An ellipsis indicates that you can repeat the preceding element.
'	Words in single quotes indicate that you must type the quotes.

mysql.rds_set_external_master

Configures a MySQL DB instance to be a Read Replica of an instance of MySQL running external to Amazon RDS.

Syntax

```
CALL mysql.rds_set_external_master (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Parameters

host_name
: The host name or IP address of the MySQL instance running external to Amazon RDS that will become the replication master.

host_port
: The port used by the MySQL instance running external to Amazon RDS to be configured as the replication master. If your network configuration includes SSH port replication that converts the port number, specify the port number that is exposed by SSH.

replication_user_name
The ID of a user with REPLICATION SLAVE permissions in the MySQL DB instance to be configured as the Read Replica.

replication_user_password
The password of the user ID specified in `replication_user_name`.

mysql_binary_log_file_name
The name of the binary log on the replication master contains the replication information.

mysql_binary_log_file_location
The location in the `mysql_binary_log_file_name` binary log at which replication will start reading the replication information.

ssl_encryption
This option is not currently implemented. The default is 0.

Usage Notes

The `mysql.rds_set_external_master` procedure must be run by the master user. It must be run on the MySQL DB instance to be configured as the Read Replica of a MySQL instance running external to Amazon RDS. Before running `mysql.rds_set_external_master`, you must have configured the instance of MySQL running external to Amazon RDS as a replication master. For more information, see Importing and Exporting Data From a MySQL DB Instance (p. 143).

Warning
Do not use `mysql.rds_set_external_master` to manage replication between two Amazon RDS DB instances. Use it only when replicating with an instance of MySQL running external to RDS. For information about managing replication between Amazon RDS DB instances, see Working with PostgreSQL and MySQL Read Replicas (p. 472).

After calling `mysql.rds_set_external_master` to configure an Amazon RDS DB instance as a Read Replica, you can call mysql.rds_start_replication (p. 180) to start the replication process. You can call mysql.rds_reset_external_master (p. 180) to remove the Read Replica configuration.

When `mysql.rds_set_external_master` is called, Amazon RDS records the time, user, and an action of "set master" in the `mysql.rds_history` and `mysql.rds_replication_status` tables.

The `mysql.rds_set_external_master` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5 version 5.5.33 or later
- MySQL 5.6 version 5.6.13 or later

Examples

When run on a MySQL DB instance, the following example configures the DB instance to be a Read Replica of an instance of MySQL running external to Amazon RDS.

```
call mysql.rds_set_external_master('Sourcedb.some.com',3306,'Replicatio
nUser','SomePassW0rd','mysql-bin-changelog.0777',120,0);
```

Related Topics

- mysql.rds_reset_external_master (p. 180)
- mysql.rds_start_replication (p. 180)
- mysql.rds_stop_replication (p. 181)

mysql.rds_reset_external_master

Reconfigures a MySQL DB instance to no longer be a Read Replica of an instance of MySQL running external to Amazon RDS.

Syntax

```
CALL mysql.rds_reset_external_master;
```

Usage Notes

The `mysql.rds_reset_external_master` procedure must be run by the master user. It must be run on the MySQL DB instance to be removed as a Read Replica of a MySQL instance running external to Amazon RDS.

> **Warning**
> Do not use `mysql.rds_reset_external_master` to manage replication between two Amazon RDS DB instances. Use it only when replicating with an instance of MySQL running external to Amazon RDS. For information about managing replication between Amazon RDS DB instances, see Working with PostgreSQL and MySQL Read Replicas (p. 472).

For more information about using replication to import data from an instance of MySQL running external to Amazon RDS, see Importing and Exporting Data From a MySQL DB Instance (p. 143).

The `mysql.rds_reset_external_master` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5 version 5.5.33 or later
- MySQL 5.6 version 5.6.13 or later

Related Topics

- mysql.rds_set_external_master (p. 178)
- mysql.rds_start_replication (p. 180)
- mysql.rds_stop_replication (p. 181)

mysql.rds_start_replication

Initiates replication from a MySQL DB instance.

Syntax

```
CALL mysql.rds_start_replication;
```

Usage Notes

The `mysql.rds_start_replication` procedure must be run by the master user.

If you are configuring replication to import data from an instance of MySQL running external to Amazon RDS, you call `mysql.rds_start_replication` to start the replication process after you have called

mysql.rds_set_external_master (p. 178) to build the replication configuration. For more information, see Importing and Exporting Data From a MySQL DB Instance (p. 143).

If you are configuring replication to export data to an instance of MySQL external to Amazon RDS, you call `mysql.rds_start_replication` and `mysql.rds_stop_replication` to control some replication actions, such as purging binary logs. For more information, see Using Replication to Export MySQL 5.6 Data (p. 164).

You can also use `mysql.rds_start_replication` to restart any replication process that you previously stopped by calling mysql.rds_stop_replication (p. 181). For more information, see Working with PostgreSQL and MySQL Read Replicas (p. 472).

The `mysql.rds_start_replication` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5 version 5.5.33 or later
- MySQL 5.6 version 5.6.13 or later

Related Topics

- mysql.rds_set_external_master (p. 178)
- mysql.rds_reset_external_master (p. 180)
- mysql.rds_stop_replication (p. 181)

mysql.rds_stop_replication

Terminates replication from a MySQL DB instance.

Syntax

```
CALL mysql.rds_stop_replication;
```

Usage Notes

The `mysql.rds_stop_replication` procedure must be run by the master user.

If you are configuring replication to import data from an instance of MySQL running external to Amazon RDS, you call `mysql.rds_stop_replication` to stop the replication process after the import has completed. For more information, see Importing and Exporting Data From a MySQL DB Instance (p. 143).

If you are configuring replication to export data to an instance of MySQL external to Amazon RDS, you call `mysql.rds_start_replication` and `mysql.rds_stop_replication` to control some replication actions, such as purging binary logs. For more information, see Using Replication to Export MySQL 5.6 Data (p. 164).

You can also use `mysql.rds_stop_replication` to stop replication between two Amazon RDS DB instances. You typically stop replication to perform a long running operation on the replica, such as creating a large index on the replica. You can restart any replication process that you stopped by calling mysql.rds_start_replication (p. 180). For more information, see Working with PostgreSQL and MySQL Read Replicas (p. 472).

The `mysql.rds_stop_replication` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5 version 5.5.33 or later
- MySQL 5.6 version 5.6.13 or later

Related Topics

- mysql.rds_set_external_master (p. 178)
- mysql.rds_reset_external_master (p. 180)
- mysql.rds_start_replication (p. 180)

mysql_rds_skip_repl_error

Skips and deletes a replication error on a MySQL DB instance.

Syntax

```
CALL mysql.rds_skip_repl_error;
```

Usage Notes

The `mysql.rds_skip_repl_error` must be run by the master user.

Run the MySQL `show slave status\G` command to determine if there are errors. If a replication error is not critical, you can elect to use `mysql.rds_skip_repl_error` to skip the error. If there are multiple errors, `mysql.rds_skip_repl_error` deletes the first error, then warns that others are present. You can then use `show slave status\G` to determine the correct course of action for the next error. For information about the values returned, go to SHOW SLAVE STATUS Syntax in the MySQL documentation.

For more information about addressing replication errors with Amazon RDS, see Troubleshooting a MySQL Read Replica Problem (p. 483).

The `mysql.rds_skip_repl_error` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.1 version 5.1.62 or later.
- MySQL 5.5 version 5.5.23 or later.
- MySQL 5.6 version 5.6.12 or later.

Important
If you attempt to call `mysql.rds_skip_repl_error` and encounter the following error: `ERROR 1305 (42000): PROCEDURE mysql.rds_skip_repl_error does not exist`, then upgrade your MySQL DB instance to the latest minor version or one of the minimum minor versions listed in this topic.

mysql.rds_next_master_log

Changes the replication master log position to the start of the next binary log on the master. Use this procedure only if you are receiving replication I/O error 1236 on a Read Replica.

Syntax

```
CALL mysql.rds_next_master_log(
curr_master_log
);
```

Parameters

curr_master_log

The index of the current master log file. For example, if the current file is named `mysql-bin-changelog.012345`, then the index is 12345. To determine the current master log file name, run the `SHOW SLAVE STATUS` command and view the `Master_Log_File` field.

Usage Notes

The `mysql.rds_next_master_log` procedure must be run by the master user.

Warning

Call `mysql.rds_next_master_log` only if replication fails after a failover of a Multi-AZ DB instance that is the replication source, and the `Last_IO_Errno` field of `SHOW SLAVE STATUS` reports I/O error 1236.

Calling `mysql.rds_next_master_log` may result in data loss in the Read Replica if transactions in the source instance were not written to the binary log on disk before the failover event occurred. You can reduce the chance of this happening by configuring the source instance parameters sync_binlog = 1 and innodb_support_xa = 1, although this may reduce performance. For more information, see Working with PostgreSQL and MySQL Read Replicas (p. 472).

The `mysql.rds_next_master_log` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.1 version 5.1.71 or later
- MySQL 5.5 version 5.5.33 or later
- MySQL 5.6 version 5.6.13 or later

Examples

Assume replication fails on an Amazon RDS Read Replica. Running `SHOW SLAVE STATUS\G` on the replica returns the following result:

```
*************************** 1. row ***************************
               Slave_IO_State:
                Master_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com

                  Master_User: MasterUser
                  Master_Port: 3306
                Connect_Retry: 10
              Master_Log_File: mysql-bin-changelog.012345
          Read_Master_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Master_Log_File: mysql-bin-changelog.012345
             Slave_IO_Running: No
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
```

```
          Exec_Master_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: NULL
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from master when reading
data from binary log: 'Client requested master to start replication from im
possible position; the first event 'mysql-bin-changelog.013406' at 1219393, the
 last event read from '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4,
 the last byte read from '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at
 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 67285976
```

The `Last_IO_Errno` field shows that the instance is receiving I/O error 1236. The `Master_Log_File` field shows that the file name is `mysql-bin-changelog.012345`, which means that the log file index is `12345`. To resolve the error, you can call `mysql.rds_next_master_log` with the following parameter:

```
CALL mysql.rds_next_master_log(12345);
```

mysql.rds_innodb_buffer_pool_dump_now

Dumps the current state of the buffer pool to disk. For more information, see InnoDB Cache Warming (p. 123).

Syntax

```
CALL mysql.rds_innodb_buffer_pool_dump_now();
```

Usage Notes

The `mysql.rds_innodb_buffer_pool_dump_now` procedure must be run by the master user.

The `mysql.rds_innodb_buffer_pool_dump_now` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6 version 5.6.19 and later

Related Topics

- mysql.rds_innodb_buffer_pool_load_now (p. 185)
- mysql.rds_innodb_buffer_pool_load_abort (p. 185)

mysql.rds_innodb_buffer_pool_load_now

Loads the saved state of the buffer pool from disk. For more information, see InnoDB Cache Warming (p. 123).

Syntax

```
CALL mysql.rds_innodb_buffer_pool_load_now();
```

Usage Notes

The `mysql.rds_innodb_buffer_pool_load_now` procedure must be run by the master user.

The `mysql.rds_innodb_buffer_pool_load_now` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6 version 5.6.19 and later

Related Topics

- mysql.rds_innodb_buffer_pool_dump_now (p. 184)
- mysql.rds_innodb_buffer_pool_load_abort (p. 185)

mysql.rds_innodb_buffer_pool_load_abort

Cancels a load of the saved buffer pool state while in progress. For more information, see InnoDB Cache Warming (p. 123).

Syntax

```
CALL mysql.rds_innodb_buffer_pool_load_abort();
```

Usage Notes

The `mysql.rds_innodb_buffer_pool_load_abort` procedure must be run by the master user.

The `mysql.rds_innodb_buffer_pool_load_abort` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6 version 5.6.19 and later

Related Topics

- mysql.rds_innodb_buffer_pool_dump_now (p. 184)
- mysql.rds_innodb_buffer_pool_load_now (p. 185)

mysql.rds_set_configuration

Specifies the number of hours to retain binary logs.

Syntax

```
CALL mysql.rds_set_configuration(name,value);
```

Parameters

name
The name of the configuration parameter to set.

value
The value of the configuration parameter.

Usage Notes

The `mysql.rds_set_configuration` procedure currently supports only the `binlog retention hours` configuration parameter. The `binlog retention hours` parameter is used to specify the number of hours to retain binary log files. Amazon RDS normally purges a binary log as soon as possible, but the binary log might still be required for replication with a MySQL database external to Amazon RDS.

To specify the number of hours for Amazon RDS to retain binary logs on a DB instance, use the `mysql.rds_set_configuration` stored procedure and specify a period with enough time for replication to occur, as shown in the following example.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

After you set the retention period, monitor storage usage for the DB instance to ensure that the retained binary logs don't take up too much storage.

The `mysql.rds_set_configuration` is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6

Related Topics

- mysql.rds_show_configuration (p. 186)

mysql.rds_show_configuration

Displays the number of hours binary logs will be retained.

Syntax

```
CALL mysql.rds_show_configuration;
```

Usage Notes

To verify the number of hours Amazon RDS will retain binary logs, use the `mysql.rds_show_configuration` stored procedure.

The `mysql.rds_show_configuration` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6

Related Topics

- mysql.rds_set_configuration (p. 186)

Examples

The following example displays the retention period:

```
call mysql.rds_show_configuration;
               name                          value     description
               binlog retention hours        24        binlog retention hours
specifies the duration in hours before binary logs are automatically deleted.
```

mysql.rds_kill

Terminates a connection to the MySQL server.

Syntax

```
CALL mysql.rds_kill(processID);
```

Parameters

processID
The identity of the connection thread that will be terminated.

Usage Notes

Each connection to the MySQL server runs in a separate thread. To terminate a connection, use the `mysql_rds_kill` procedure and pass in the thread ID of that connection. To obtain the thread ID, use the MySQL SHOW PROCESSLIST command.

The `mysql.rds_kill` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6

Related Topics

- mysql.rds_kill_query (p. 188)

Examples

The following example terminates a connection with a thread ID of 4243:

```
call mysql.rds_kill(4243);
```

mysql.rds_kill_query

Terminates a query running against the MySQL server.

Syntax

```
CALL mysql.rds_kill_query(queryID);
```

Parameters

queryID
The identity of the query that will be terminated.

Usage Notes

To terminate a query running against the MySQL server, use the `mysql_rds_kill_query` procedure and pass in the ID of that query. To obtain the query ID, use the MySQL INFORMATION_SCHEMA PROCESSLIST command. The connection to the MySQL server will be retained.

The `mysql_rds_kill_query` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6

Related Topics

- mysql.rds_kill (p. 187)

Examples

The following example terminates a query with a thread ID of 230040:

```
call mysql.rds_kill_query(230040);
```

mysql.rds_rotate_general_log

Rotates the `mysql.general_log` table to a backup table. For more information, see MySQL Database Log Files (p. 594).

Syntax

```
CALL mysql.rds_rotate_general_log;
```

Usage Notes

You can rotate the `mysql.general_log` table to a backup table by calling the `mysql.rds_rotate_general_log` procedure. When log tables are rotated, the current log table is copied to a backup log table and the entries in the current log table are removed. If a backup log table already exists, then it is deleted before the current log table is copied to the backup. You can query the backup log table if needed. The backup log table for the `mysql.general_log` table is named `mysql.general_log_backup`.

The `mysql.rds_rotate_general_log` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6

Related Topics

- mysql.rds_rotate_slow_log (p. 189)

mysql.rds_rotate_slow_log

Rotates the `mysql.slow_log` table to a backup table. For more information, see MySQL Database Log Files (p. 594).

Syntax

```
CALL mysql.rds_rotate_slow_log;
```

Usage Notes

You can rotate the `mysql.slow_log` table to a backup table by calling the `mysql.rds_rotate_slow_log` procedure. When log tables are rotated, the current log table is copied to a backup log table and the entries in the current log table are removed. If a backup log table already exists, then it is deleted before the current log table is copied to the backup.

You can query the backup log table if needed. The backup log table for the `mysql.slow_log` table is named `mysql.slow_log_backup`.

The `mysql.rds_rotate_slow_log` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6

Related Topics

- mysql.rds_rotate_general_log (p. 188)

mysql.rds_enable_gsh_collector

Enables the Global Status History (GoSH) to take default snapshots at intervals specified by `rds_set_gsh_collector`. For more information, see Managing the Global Status History (p. 171).

Syntax

```
CALL mysql.rds_enable_gsh_collector;
```

Related Topics

- mysql.rds_set_gsh_collector (p. 190)
- mysql.rds_disable_gsh_collector (p. 191)
- mysql.rds_collect_global_status_history (p. 191)
- mysql.rds_enable_gsh_rotation (p. 191)
- mysql.rds_set_gsh_rotation (p. 192)
- mysql.rds_disable_gsh_rotation (p. 192)
- mysql.rds_rotate_global_status_history (p. 193)

mysql.rds_set_gsh_collector

Specifies the interval, in minutes, between snapshots taken by the Global Status History (GoSH). Default value is 5. For more information, see Managing the Global Status History (p. 171).

Syntax

```
CALL mysql.rds_set_gsh_collector(intervalPeriod);
```

Parameters

intervalPeriod
The interval, in minutes, between snapshots. Default value is 5.

Related Topics

- mysql.rds_enable_gsh_collector (p. 190)
- mysql.rds_disable_gsh_collector (p. 191)
- mysql.rds_collect_global_status_history (p. 191)
- mysql.rds_enable_gsh_rotation (p. 191)
- mysql.rds_set_gsh_rotation (p. 192)
- mysql.rds_disable_gsh_rotation (p. 192)
- mysql.rds_rotate_global_status_history (p. 193)

mysql.rds_disable_gsh_collector

Disables snapshots taken by the Global Status History (GoSH). For more information, see Managing the Global Status History (p. 171).

Syntax

```
CALL mysql.rds_disable_gsh_collector;
```

Related Topics

- mysql.rds_enable_gsh_collector (p. 190)
- mysql.rds_set_gsh_collector (p. 190)
- mysql.rds_collect_global_status_history (p. 191)
- mysql.rds_enable_gsh_rotation (p. 191)
- mysql.rds_set_gsh_rotation (p. 192)
- mysql.rds_disable_gsh_rotation (p. 192)
- mysql.rds_rotate_global_status_history (p. 193)

mysql.rds_collect_global_status_history

Takes a snapshot on demand for the Global Status History (GoSH). For more information, see Managing the Global Status History (p. 171).

Syntax

```
CALL rds.collect_global_status_history;
```

Related Topics

- mysql.rds_enable_gsh_collector (p. 190)
- mysql.rds_set_gsh_collector (p. 190)
- mysql.rds_disable_gsh_collector (p. 191)
- mysql.rds_enable_gsh_rotation (p. 191)
- mysql.rds_set_gsh_rotation (p. 192)
- mysql.rds_disable_gsh_rotation (p. 192)
- mysql.rds_rotate_global_status_history (p. 193)

mysql.rds_enable_gsh_rotation

Enables rotation of the contents of the `mysql.global_status_history` table to `mysql.global_status_history_old` at intervals specified by `rds_set_gsh_rotation`. For more information, see Managing the Global Status History (p. 171).

Syntax

```
CALL mysql.rds_enable_gsh_rotation;
```

Related Topics

- mysql.rds_enable_gsh_collector (p. 190)
- mysql.rds_set_gsh_collector (p. 190)
- mysql.rds_disable_gsh_collector (p. 191)
- mysql.rds_collect_global_status_history (p. 191)
- mysql.rds_set_gsh_rotation (p. 192)
- mysql.rds_disable_gsh_rotation (p. 192)
- mysql.rds_rotate_global_status_history (p. 193)

mysql.rds_set_gsh_rotation

Specifies the interval, in days, between rotations of the `mysql.global_status_history` table. Default value is 7. For more information, see Managing the Global Status History (p. 171).

Syntax

```
CALL mysql.rds_set_gsh_rotation(intervalPeriod);
```

Parameters

intervalPeriod
The interval, in days, between table rotations. Default value is 7.

Related Topics

- mysql.rds_enable_gsh_collector (p. 190)
- mysql.rds_set_gsh_collector (p. 190)
- mysql.rds_disable_gsh_collector (p. 191)
- mysql.rds_collect_global_status_history (p. 191)
- mysql.rds_enable_gsh_rotation (p. 191)
- mysql.rds_disable_gsh_rotation (p. 192)
- mysql.rds_rotate_global_status_history (p. 193)

mysql.rds_disable_gsh_rotation

Disables rotation of the `mysql.global_status_history` table. For more information, see Managing the Global Status History (p. 171).

Syntax

```
CALL mysql.rds_disable_gsh_rotation;
```

Related Topics

- mysql.rds_enable_gsh_collector (p. 190)
- mysql.rds_set_gsh_collector (p. 190)
- mysql.rds_disable_gsh_collector (p. 191)
- mysql.rds_collect_global_status_history (p. 191)
- mysql.rds_enable_gsh_rotation (p. 191)
- mysql.rds_set_gsh_rotation (p. 192)
- mysql.rds_rotate_global_status_history (p. 193)

mysql.rds_rotate_global_status_history

Rotates the contents of the `mysql.global_status_history` table to `mysql.global_status_history_old` on demand. For more information, see Managing the Global Status History (p. 171).

Syntax

```
CALL mysql.rds_rotate_global_status_history;
```

Related Topics

- mysql.rds_enable_gsh_collector (p. 190)
- mysql.rds_set_gsh_collector (p. 190)
- mysql.rds_disable_gsh_collector (p. 191)
- mysql.rds_collect_global_status_history (p. 191)
- mysql.rds_enable_gsh_rotation (p. 191)
- mysql.rds_set_gsh_rotation (p. 192)
- mysql.rds_disable_gsh_rotation (p. 192)

Oracle on Amazon RDS

Amazon RDS supports DB instances running one of several editions of Oracle Database. You can create DB instances and DB snapshots, point-in-time restores and automated or manual backups. DB instances running Oracle can be used inside a VPC. You can also enable various options to add additional features to your Oracle DB instance. Amazon RDS currently supports Multi-AZ deployments for Oracle as a high-availability, failover solution.

In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application such as Oracle SQL Plus. Amazon RDS does not allow direct host access to a DB instance via Telnet or Secure Shell (SSH). When you create a DB instance, you create a master account that gets DBA privileges (with some limitations) and the SYS password or SYSDBA privileges are not provided.

These are the common management tasks you perform with an Amazon RDS Oracle DB instance, with links to information about each task:

- For planning information, such as Oracle versions, storage engines, security, and features supported in Amazon RDS, see Planning Your Amazon RDS Oracle DB Instance (p. 195).
- Before creating a DB instance, you should complete the steps in the Setting Up for Amazon RDS (p. 7) section of this guide.
- If you are creating a DB instance for production purposes, you should understand how instance classes, storage, and Provisioned IOPS work in Amazon RDS. For more information about DB instance classes, see DB Instance Class (p. 65) For more information about Amazon RDS storage, see Amazon RDS Storage Types (p. 77). For more information about Provisioned IOPS, see Amazon RDS Provisioned IOPS Storage to Improve Performance (p. 82).
- A production DB instance should also use Multi-AZ deployments. All Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances. For more information about Multi-AZ deployments, see High Availability (Multi-AZ) (p. 71).
- There are prerequisites you must complete before you create your DB instance. For example, DB instances are created by default with a firewall that prevents access to it. You therefore must create a security group with the correct IP addresses and network configuration you will use to access the DB instance. The security group you need to create will depend on what EC2 platform your DB instance is on, and whether you will be accessing your DB instance from an EC2 instance. For more information about the two EC2 platforms supported by Amazon RDS, *EC2-VPC* and *EC2-Classic*, see Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 563). In general, if your DB instance is on the *EC2-Classic* platform, you will need to create a DB security group; if your DB instance is on the *EC2-VPC* platform, you will need to create a VPC security group. For more information about

security groups, see Amazon RDS Security Groups (p. 110) or the Setting Up for Amazon RDS (p. 7) section of this guide.

- If your AWS account has a default VPC (a default virtual private network), then your DB instance will automatically be created inside the default VPC. If your account does not have a default VPC and you want the DB instance to be inside a VPC, you must create the VPC and subnet groups before you create the DB instance. For more information about determining if your account has a default VPC, see Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 563). For more information about using VPCs with Amazon RDS, see Using Amazon RDS with Amazon Virtual Private Cloud (VPC) (p. 562).
- If your DB instance is going to require specific database parameters or options, you should create the parameter or option groups before you create the DB instance. For more information on parameter groups, see Working with DB Parameter Groups (p. 523). For more information on options for Oracle, see Appendix: Options for Oracle Database Engine (p. 226).
- After creating a security group and associating it to a DB instance, you can connect to the DB instance using any standard SQL client application such as Oracle SQL Plus. For more information on connecting to a DB instance, see Connecting to a DB Instance Running the Oracle Database Engine (p. 212).
- You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots. For information, see Backing Up and Restoring (p. 495).
- You can monitor an instance through actions such as viewing the Oracle logs, CloudWatch Amazon RDS metrics, and events. For information, see Monitoring Amazon RDS (p. 571).

There are also several appendices with useful information about working with Oracle DB instances:

- For information on common DBA tasks for Oracle on Amazon RDS, see Appendix: Common DBA Tasks for Oracle (p. 241).
- For information on the options that you can use with Oracle on Amazon RDS, see Appendix: Options for Oracle Database Engine (p. 226).

Planning Your Amazon RDS Oracle DB Instance

Amazon RDS supports DB instances running several editions of Oracle Database. This section shows how you can work with Oracle on Amazon RDS. You should also be aware of the limits for Oracle DB instances.

For information about importing Oracle data into a DB instance, see Importing Data Into Oracle on Amazon RDS (p. 219).

Topics

- Oracle Database Engine Options (p. 195)
- Security (p. 203)
- Oracle Version Management (p. 203)
- Licensing (p. 203)
- Using OEM, APEX, TDE, and other options (p. 204)

Oracle Database Engine Options

The following list shows a subset of the key Oracle database engine features that are currently supported by Amazon RDS. The availability of the Oracle feature is dependent on the edition of Oracle that you choose. For example, OEM optional packs such as the Database Diagnostic Pack and the Database Tuning Pack are only available with Oracle Enterprise Edition.

Oracle 12c with Amazon RDS

Oracle version 12c brings over 500 new features and updates from the previous version. This section covers the features and changes important to using Oracle 12c on Amazon RDS. For a complete list of the changes, see the Oracle 12c documentation.

Oracle 12c includes sixteen new parameters that impact your Amazon RDS DB instance, as well as eighteen new system privileges, several no longer supported packages, and several new option group settings. The following sections provide more information on these changes.

Amazon RDS Parameter Changes for Oracle 12c

Oracle 12c includes sixteen new parameters in addition to several parameters with new ranges and new default values.

The following table shows the new Amazon RDS parameters for Oracle 12c:

Name	Values	Modifiable	Description
connection_brokers	CONNECTION_BROKERS = broker_description[,...]	N	Specifies connection broker types, the number of connection brokers of each type, and the maximum number of connections per broker.
db_index_compression_inheritance	TABLESPACE, TABL, ALL, NONE	Y	Displays the options that are set for table or tablespace level compression inheritance.
db_big_table_cache_percent_target	0-90	Y	Specifies the cache section target size for automatic big table caching, as a percentage of the buffer cache.
heat_map	ON,OFF	Y	Enables the database to track read and write access of all segments, as well as modification of database blocks, due to DMLs and DDLs.
inmemory_clause_default	INMEMORY,NO INMEMORY	Y	INMEMORY_CLAUSE_DEFAULT enables you to specify a default In-Memory Column Store (IM column store) clause for new tables and materialized views.

Name	Values	Modifiable	Description
inmemory_clause_default_memcompress	NO MEMCOMPRESS,MEMCOMPRESS FOR DML,MEMCOMPRESS FOR QUERY, MEMCOMPRESS FOR QUERY LOW,MEMCOMPRESS FOR QUERY HIGH,MEMCOMPRESS FOR CAPACITY,MEMCOMPRESS FOR CAPACITY LOW,MEMCOMPRESS FOR CAPACITY HIGH	Y	See INMEMORY_CLAUSE_DEFAULT.
inmemory_clause_default_priority	PRIORITY LOW,PRIORITY MEDIUM,PRIORITY HIGH,PRIORITY CRITICAL,PRIORITY NONE	Y	See INMEMORY_CLAUSE_DEFAULT.
inmemory_force	DEFAULT, OFF	Y	INMEMORY_FORCE allows you to specify whether tables and materialized view that are specified as INMEMORY are populated into the In-Memory Column Store (IM column store) or not.
inmemory_max_populate_servers	Null	N	INMEMORY_MAX_POPULATE_SERVERS specifies the maximum number of background populate servers to use for In-Memory Column Store (IM column store) population, so that these servers do not overload the rest of the system.
inmemory_query	ENABLE (default), DISABLE	Y	INMEMORY_QUERY is used to enable or disable in-memory queries for the entire database at the session or system level.
inmemory_size	0,104857600-274877906944	Y	INMEMORY_SIZE sets the size of the In-Memory Column Store (IM column store) on a database instance.
inmemory_trickle_repopulate_servers_percent	0 to 50	Y	INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT limits the maximum number of background populate servers used for In-Memory Column Store (IM column store) repopulation, as trickle repopulation is designed to use only a small percentage of the populate servers.
max_string_size	STANDARD (default), EXTENDED	N	Controls the maximum size of VARCHAR2, NVARCHAR2, and RAW.

Name	Values	Modifiable	Description
optimizer_adaptive_features	TRUE (default), FALSE	Y	Enables or disables all of the adaptive optimizer features.
optimizer_adaptive_reporting_only	TRUE,FALSE (default)	Y	Controls reporting-only mode for adaptive optimizations.
pdb_file_name_convert		N	Maps names of existing files to new file names.
pga_aggregate_limit	1-max of memory	Y	Specifies a limit on the aggregate PGA memory consumed by the instance.
processor_group_name		N	Instructs the database instance to run itself within the specified operating system processor group.
spatial_vector_acceleration	TRUE,FALSE	N	Enables or disables the spatial vector acceleration, part of spacial option.
temp_undo_enabled	TRUE,FALSE (default)	Y	Determines whether transactions within a particular session can have a temporary undo log.
threaded_execution	TRUE,FALSE	N	Enables the multithreaded Oracle model, but prevents OS authentication.
unified_audit_sga_queue_size	1 MB - 30 MB	Y	Specifies the size of SGA queue for unified auditing.
use_dedicated_broker	TRUE,FALSE	N	Determines how dedicated servers are spawned.

Several parameter have new value ranges for Oracle 12c on Amazon RDS. The following table shows the old and new value ranges:

Parameter Name	12c Range	11g Range
audit_trail	os | db [, extended] | xml [, extended]	
compatible	Starts with 11.0.0	Starts with 10.0.0
db_securefile	PERMITTED | PREFERRED | ALWAYS | IGNORE | FORCE	PERMITTED | ALWAYS | IGNORE | FORCE
db_writer_processes	1-100	1-36
optimizer_features_enable	8.0.0 to 12.1.0.1	8.0.0 to 11.2.0.1
parallel_degree_policy	MANUAL,LIMITED,AUTO,ADAPTIVE	MANUAL,LIMITED,AUTO
parallel_min_server	0 to parallel_max_servers	CPU_COUNT * PARALLEL_THREADS_PER_CPU * 2 to parallel_max_servers

One parameters has a new default value for Oracle 12c on Amazon RDS. The following table shows the new default value:

Parameter Name	Oracle 12c Default Value	Oracle 11g Default Value
job_queue_processes	50	1000

Amazon RDS System Privileges for Oracle 12c

Several new system privileges have been granted to the system account for Oracle 12c. These new system privileges include:

- ALTER ANY CUBE BUILD PROCESS
- ALTER ANY MEASURE FOLDER
- ALTER ANY SQL TRANSLATION PROFILE
- CREATE ANY SQL TRANSLATION PROFILE
- CREATE SQL TRANSLATION PROFILE
- DROP ANY SQL TRANSLATION PROFILE
- EM EXPRESS CONNECT
- EXEMPT DDL REDACTION POLICY
- EXEMPT DML REDACTION POLICY
- EXEMPT REDACTION POLICY
- LOGMINING
- REDEFINE ANY TABLE
- SELECT ANY CUBE BUILD PROCESS
- SELECT ANY MEASURE FOLDER
- USE ANY SQL TRANSLATION PROFILE

Amazon RDS Options for Oracle 12c

Several Oracle option changed between Oracle 11g and Oracle 12c, though most of the options remain the same between the two versions. The Oracle 12c changes include:

- Oracle Enterprise Manager Express (EM Express) replaced Oracle Enterprise Manager DB Control. For more information see Oracle Database 12c: EM Database Express.
- The option XMLDB is installed by default in Oracle 12c. It is no longer an option that you need to install.
- The Oracle APEX Listener has been renamed to Oracle Rest Data Service (ORDS). ORDS is installed on a separate EC2 instance just as the APEX Listener was in version 11g. The process for installing ORDS is not the same as when installing APEX Listener. For instructions on installing ORDS, see Oracle APEX on Amazon RDS Oracle 12c (p. 230).
- APEX and APEX Dev no longer have a dependency on XMLDB since XMLDB is installed by default.

Amazon RDS PL/SQL Packages for Oracle 12c

Oracle 12c includes a number of new built-in PL/SQL packages. The packages included with Amazon RDS Oracle 12c include the following:

Package Name	Description
CTX_ANL	The CTX_ANL package is used with AUTO_LEXER and provides procedures for adding and dropping a custom dictionary from the lexer.
DBMS_APP_CONT	The DBMS_APP_CONT package provides an interface to determine if the in-flight transaction on a now unavailable session committed or not, and if the last call on that session completed or not.
DBMS_AUTO_REPORT	The DBMS_AUTO_REPORT package provides an interface to view SQL Monitoring and Real-time Automatic Database Diagnostic Monitor (ADDM) data that has been captured into Automatic Workload Repository (AWR).
DBMS_GOLDENGATE_AUTH	The DBMS_GOLDENGATE_AUTH package provides subprograms for granting privileges to and revoking privileges from GoldenGate administrators.
DBMS_HEAT_MAP	The DBMS_HEAT_MAP package provides an interface to externalize heatmaps at various levels of storage including block, extent, segment, object and tablespace.
DBMS_ILM	The DBMS_ILM package provides an interface for implementing Information Lifecycle Management (ILM) strategies using Automatic Data Optimization (ADO) policies.
DBMS_ILM_ADMIN	The DBMS_ILM_ADMIN package provides an interface to customize Automatic Data Optimization (ADO) policy execution.
DBMS_PART	The DBMS_PART package provides an interface for maintenance and management operations on partitioned objects.
DBMS_PRIVILEGE_CAPTURE	The DBMS_PRIVILEGE_CAPTURE package provides an interface to database privilege analysis.
DBMS_QOPATCH	The DBMS_QOPATCH package provides an interface to view the installed database patches.
DBMS_REDACT	The DBMS_REDACT package provides an interface to Oracle Data Redaction, which enables you to mask (redact) data that is returned from queries issued by low-privileged users or an application.
DBMS_SPD	The DBMS_SPD package provides subprograms for managing SQL plan directives (SPD).
DBMS_SQL_TRANSLATOR	The DBMS_SQL_TRANSLATOR package provides an interface for creating, configuring, and using SQL translation profiles.
DBMS_SQL_MONITOR	The DBMS_SQL_MONITOR package provides information about real-time SQL Monitoring and real-time Database Operation Monitoring.
DBMS_SYNC_REFRESH	The DBMS_SYNC_REFRESH package provides an interface to perform a synchronous refresh of materialized views.

Package Name	Description
DBMS_TSDP_MANAGE	The DBMS_TSDP_MANAGE package provides an interface to import and manage sensitive columns and sensitive column types in the database, and is used in conjunction with the DBMS_TSDP_PROTECT package with regard to transparent sensitive data protection (TSDP) policies. DBMS_TSDP_MANAGE is available with the Enterprise Edition only.
DBMS_TSDP_PROTECT	The DBMS_TSDP_PROTECT package provides an interface to configure transparent sensitive data protection (TSDP) policies in conjunction with the DBMS_TSDP_MANAGE package. DBMS_TSDP_PROTECT is available with the Enterprise Edition only.
DBMS_XDB_CONFIG	The DBMS_XDB_CONFIG package provides an interface for configuring Oracle XML DB and its repository.
DBMS_XDB_CONSTANTS	The DBMS_XDB_CONSTANTS package provides an interface to commonly used constants. Users should use constants instead of dynamic strings to avoid typographical errors.
DBMS_XDB_REPOS	The DBMS_XDB_REPOS package provides an interface to operate on the Oracle XML database Repository.
DBMS_XMLSCHEMA_ANNOTATE	The DBMS_XMLSCHEMA_ANNOTATE package provides an interface to manage and configure the structured storage model, mainly through the use of pre-registration schema annotations.
DBMS_XMLSTORAGE_MANAGE	The DBMS_XMLSTORAGE_MANAGE package provides an interface to manage and modify XML storage after schema registration has been completed.
DBMS_XSTREAM_ADM	The DBMS_XSTREAM_ADM package provides interfaces for streaming database changes between an Oracle database and other systems. XStream enables applications to stream out or stream in database changes.
DBMS_XSTREAM_AUTH	The DBMS_XSTREAM_AUTH package provides subprograms for granting privileges to and revoking privileges from XStream administrators.
UTL_CALL_STACK	The UTL_CALL_STACK package provides an interface to provide information about currently executing subprograms.

The following features are not supported for Oracle 12c on Amazon RDS:

- Real Application Clusters (RAC)
- Data Guard / Active Data Guard
- Cloud Control (called Oracle Enterprise Manager Grid Control in previous Oracle versions)
- Automated Storage Management
- Database Vault
- Java Support
- Locator
- Spatial

Several Oracle 11g PL/SQL packages are not supported in Oracle 12c. These packages include:

- DBMS_AUTO_TASK_IMMEDIATE
- DBMS_CDC_PUBLISH
- DBMS_CDC_SUBSCRIBE
- DBMS_EXPFIL
- DBMS_OBFUSCATION_TOOLKIT
- DBMS_RLMGR
- SDO_NET_MEM

Oracle 11g with Amazon RDS

The following list shows the Oracle 11g features supported by Amazon RDS; for a complete list of features supported by each Oracle 11g edition, go to Oracle Database 11g Editions.

- Total Recall
- Flashback Table, Query and Transaction Query
- Virtual Private Database
- Fine-Grained Auditing
- Comprehensive support for Microsoft .NET, OLE DB, and ODBC
- Automatic Memory Management
- Automatic Undo Management
- Advanced Compression
- Partitioning
- Star Query Optimization
- Summary Management - Materialized View Query Rewrite
- Oracle Data Redaction (version 11.2.0.4 or later)
- Distributed Queries/Transactions
- Text
- Materialized Views
- Import/Export and sqlldr Support
- Oracle Enterprise Manager Database Control
- Oracle XML DB (without the XML DB Protocol Server)
- Oracle Application Express
- Automatic Workload Repository for Enterprise Edition (AWR). For more information, see Working with Automatic Workload Repository (AWR) (p. 251)
- Datapump (network only)
- Native network encryption (part of the Oracle Advanced Security feature)
- Transparent data encryption (Oracle TDE, part of the Oracle Advanced Security feature)

Oracle database engine features that are not currently supported include the following:

- Real Application Clusters (RAC)
- Real Application Testing
- Data Guard / Active Data Guard
- Oracle Enterprise Manager Grid Control
- Automated Storage Management
- Database Vault
- Streams

- Java Support
- Locator
- Spatial
- Oracle XML DB Protocol Server
- Network access utilities such as utl_http, utl_tcp, utl_smtp, and utl_mail, are not supported at this time.

Security

The Oracle database engine uses role-based security. A *role* is a collection of privileges that can be granted to or revoked from a user. A predefined role, named *DBA*, normally allows all administrative privileges on an Oracle database engine. The following privileges are not available for the DBA role on an Amazon RDS DB instance using the Oracle engine:

- Alter database
- Alter system
- Create any directory
- Drop any directory
- Grant any privilege
- Grant any role

While Amazon RDS Oracle does not support SSL/TLS encrypted connections, you can use the Oracle Native Network Encryption option to encrypt connections between your application and your Oracle DB instance. For more information about the Oracle Native Network Encryption option, see Oracle Native Network Encryption (p. 233).

Oracle Version Management

DB Engine Version Management is a feature of Amazon RDS that enables you to control when and how the database engine software running your DB instances is patched and upgraded. This feature gives you the flexibility to maintain compatibility with database engine patch versions, test new patch versions to ensure they work effectively with your application before deploying in production, and perform version upgrades on your own terms and timelines.

> **Note**
> Amazon RDS periodically aggregates official Oracle database patches using an Amazon RDS-specific DB Engine version. To see a list of which Oracle patches are contained in an Amazon RDS Oracle-specific engine version, go to Appendix: Oracle Database Engine Release Notes (p. 286).

Taking advantage of the DB Engine Version Management feature of Amazon RDS is easily accomplished using the **ModifyDBInstance** API call or the **rds-modify-db-instance** command line utility. Your DB instances are upgraded to minor patches by default (you can override this setting).

Licensing

There are two types of licensing options available for using Amazon RDS for Oracle.

Bring Your Own License (BYOL)

In this licensing model, you can use your existing Oracle Database licenses to run Oracle deployments on Amazon RDS. To run a DB instance under the BYOL model, you must have the appropriate Oracle Database license (with Software Update License and Support) for the DB instance class and Oracle

Database edition you wish to run. You must also follow Oracle's policies for licensing Oracle Database software in the cloud computing environment. For more information on Oracle's licensing policy for Amazon EC2, go to Licensing Oracle Software in the Cloud Computing Environment.

License Included

In the *License Included* service model, you do not need separately purchased Oracle licenses; AWS holds the license for the Oracle Database software.

Oracle Licensing and Amazon RDS

Amazon RDS currently supports the following Oracle Database Editions under each of the licensing models below:

- BYOL: Standard Edition One (SE1), Standard Edition (SE) and Enterprise Edition (EE)

  To run a DB instance under the BYOL model, you must have the appropriate Oracle Database license (with Software Update License & Support) for the DB instance class and Oracle Database edition you wish to run. You must follow Oracle's policies for licensing Oracle Database software in the cloud computing environment. DB instances reside in the Amazon EC2 environment, and Oracle's licensing policy for Amazon EC2 is located here.

  Under this model, you will continue to use your active Oracle support account and contact Oracle directly for Oracle Database specific service requests. If you have an active AWS Premium Support account, you can contact AWS Premium Support for Amazon RDS specific issues. Amazon Web Services and Oracle have multi-vendor support process for cases which require assistance from both organizations.
- License Included: Standard Edition One (SE1)

  In the "License Included" service model, you do not need separately purchased Oracle licenses; the Oracle Database software has been licensed by AWS.

  In this model, if you have an active AWS Premium Support account, you should contact AWS Premium Support for both Amazon RDS and Oracle Database specific service requests.

Using OEM, APEX, TDE, and other options

Most Amazon RDS DB engines support option groups that allow you to select additional features for your DB instance. Oracle DB instances support several options, including OEM, TDE, APEX, and Native Network Encryption. For a complete list of supported Oracle options, see Appendix: Options for Oracle Database Engine (p. 226). For more information about working with option groups, see Working with Option Groups (p. 510).

Creating a DB Instance Running the Oracle Database Engine

The basic building block of Amazon RDS is the DB instance. This is the environment in which you will use to run your Oracle databases.

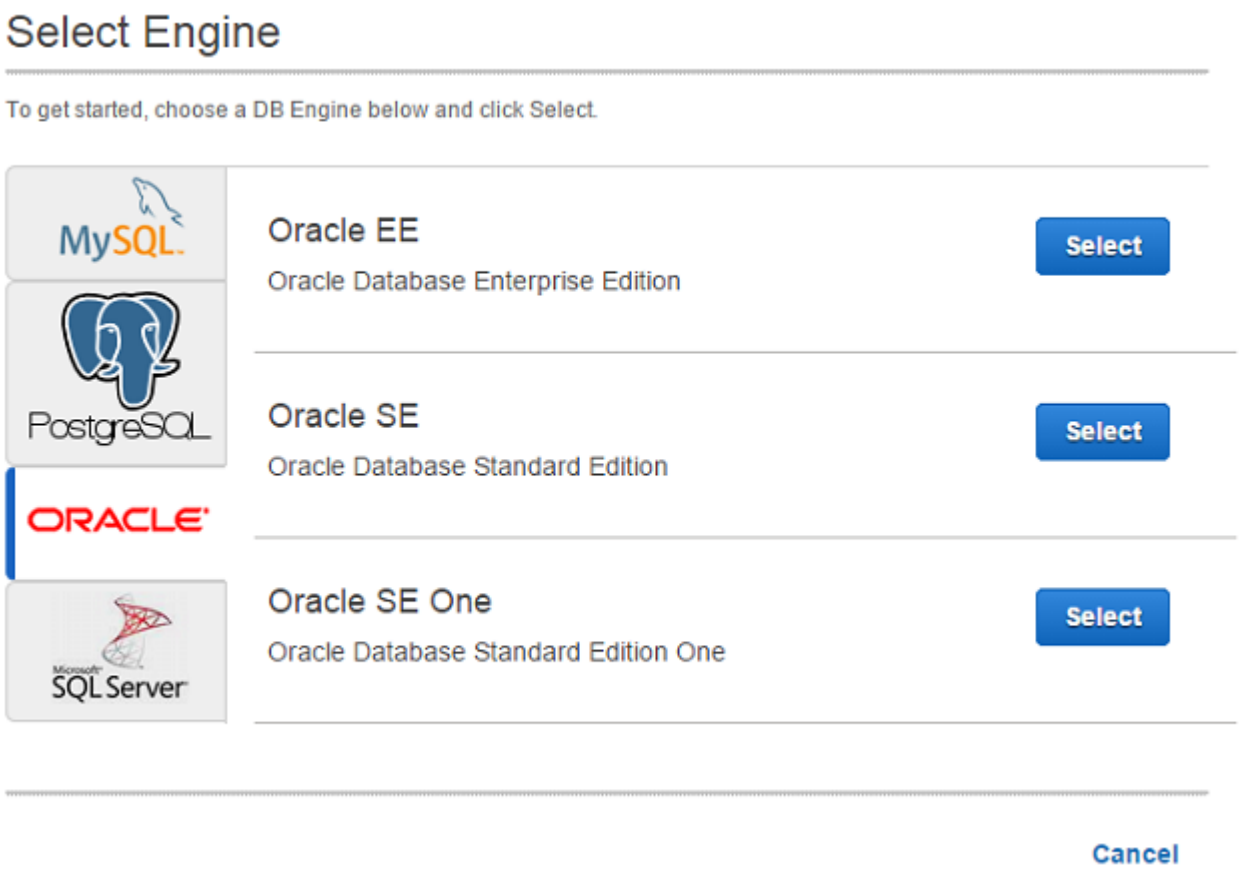
Important
You must complete the tasks in the Setting Up for Amazon RDS (p. 7) section before you can create or connect to a DB instance.

AWS Management Console

To launch an Oracle DB instance

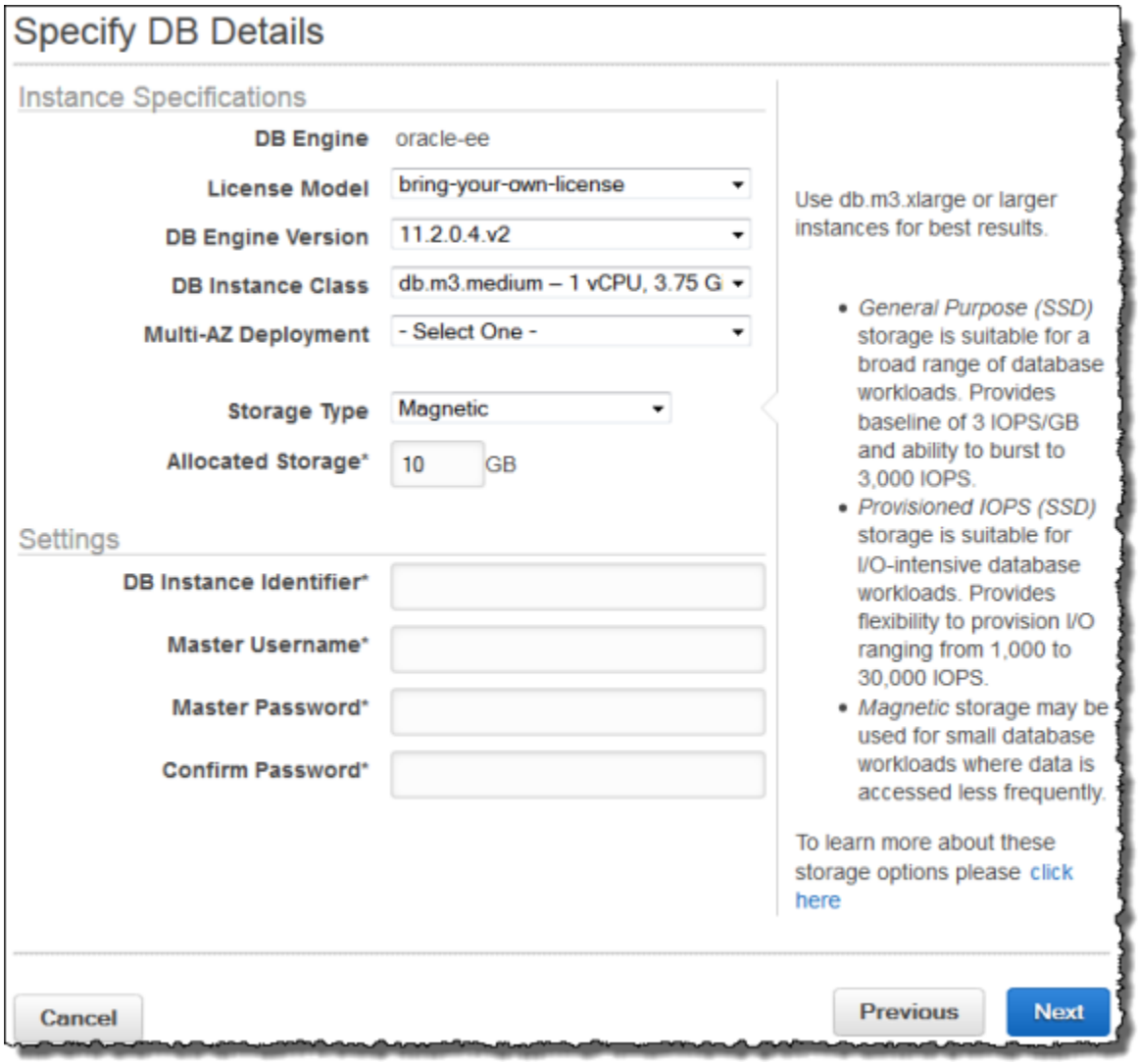
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the top right corner of the AWS Management Console, select the region in which you want to create the DB instance.
3. In the navigation pane, click **DB Instances**.
4. Click **Launch DB Instance** to start the **Launch DB Instance Wizard**.

   The wizard opens on the **Select Engine** page. The Oracle editions available will vary by region.



5. In the **Select Engine** window, click the **Select** button for the Oracle DB engine you want to use.
6. The next step asks if you are planning to use the DB instance you are creating for production. If you are, select **Yes**. By selecting **Yes**, the failover option **Multi-AZ** and the **Provisioned IOPS** storage option will be preselected in the following step. Click **Next Step** when you are finished.
7. On the **Specify DB Details** page, specify your DB instance information. The following table shows the parameters you need to set to create a DB instance. Click **Next** when you are finished.

For this parameter...	...Do this:
License Model	Select the license option you want to use. Some regions support additional licensing options for Oracle.
DB Engine Version	Select the Oracle version you want to use.
DB Instance Class	Select the DB instance class you want to use. For more information about all the DB instance class options, see DB Instance Class (p. 65).
Multi-AZ Deployment	Determine if you want to create a standby replica of your DB instance in another availability zone for failover support. This feature is available for Oracle and MySQL DB instances. For more information about multiple availability zones, see Regions and Availability Zones (p. 69).
Allocated Storage	Type a value to allocate of storage for your database (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon RDS Storage Types (p. 77).
Storage Type	Select the storage type you want to use. For more information about storage, see Storage for Amazon RDS (p. 77).
DB Instance Identifier	Type a name for the DB instance that is unique for your account in the region you selected. You may choose to add some intelligence to the name such as including the region and DB engine you selected, for example **`oracle-instance1`**.
Master User Name	Type a name that you will use as the master user name to log on to your DB instance with all database privileges. This user account is used to log into the DB instance and is granted DBA privileges.
Master User Password and **Confirm Password**	Type a password that contains from 8 to 30 printable ASCII characters (excluding /,", and @) for your master user password. Retype the password in the **Confirm Password** text box.


Specify DB Details
Instance Specifications
DB Engine
oracle-ee
License Model
bring-your-own-license
DB Engine Version
11.2.0.4.v2
DB Instance Class
db.m3.medium – 1 vCPU, 3.75 G
Multi-AZ Deployment
- Select One -
Storage Type
Magnetic
Allocated Storage*
10
GB
Settings
DB Instance Identifier*
Master Username*
Master Password*
Confirm Password*
Use db.m3.xlarge or larger instances for best results.
General Purpose (SSD) storage is suitable for a broad range of database workloads. Provides baseline of 3 IOPS/GB and ability to burst to 3,000 IOPS.
Provisioned IOPS (SSD) storage is suitable for I/O-intensive database workloads. Provides flexibility to provision I/O ranging from 1,000 to 30,000 IOPS.
Magnetic storage may be used for small database workloads where data is accessed less frequently.
To learn more about these storage options please click here
Cancel
Previous
Next

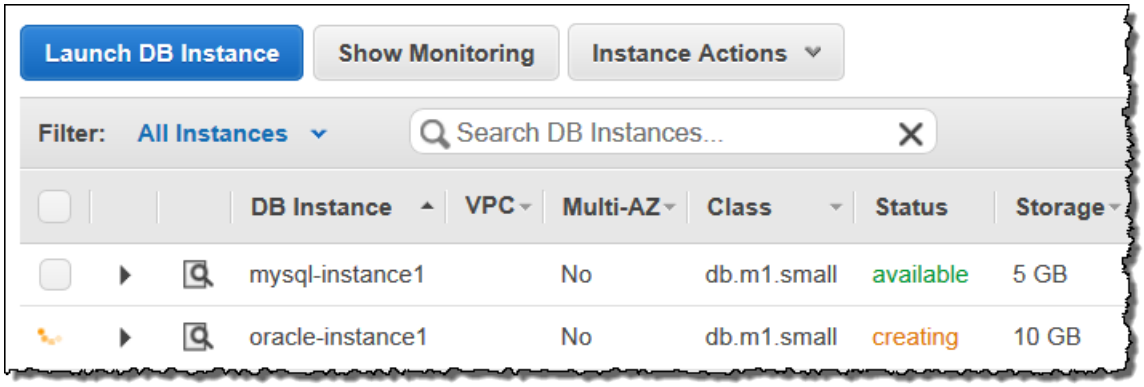
8. On the **Configure Advanced Settings** page, you provide additional information that RDS needs to launch the Oracle DB instance. The following table shows the additional parameters you provide for a DB instance. Specify your DB instance information, then click Launch DB Instance.

For this parameter...	...Do this:
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, select the default VPC. If you are creating a DB instance on the previous E2-Classic platform, select **`Not in VPC`**. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).

For this parameter...	...Do this:
DB Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, select **`default`**, which will be the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, select the DB subnet group you created for that VPC. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).
Publicly Accessible	Select **`Yes`** to give the DB instance a public IP address, meaning that it will be accessible outside the VPC; otherwise, select **`No`**, so the DB instance will only be accessible from inside the VPC. For more information about hiding DB instances from public access, see Hiding a DB instance in a VPC from the Internet.
Availability Zone	Use the default of **`No Preference`** unless you need to specify a particular Availability Zone.
VPC Security Group	If you are a new customer to AWS, select the default VPC. If you have created your own VPC security group, select the VPC security group you previously created.
Database Name	Type a name for your database that begins with a letter and contains up to 8 alpha-numeric characters. If you do not provide a name, Amazon RDS will not create a database on the DB instance you are creating.
Database Port	Specify the port you want to access the database through. Oracle installations default to port 1521.
Parameter Group	Select a parameter group. You can choose the default parameter group or you can create a parameter group and select that parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 523).
Option Group	Select an option group. You can choose the default option group or you can create an option group and select that option group. For more information about option groups, see Working with Option Groups (p. 510).
Enable Encryption	Select `Yes` to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 106).
Character Set Name	Select a character set for your DB instance. The default value of **`AL32UTF8`** is for the Unicode 5.0 UTF-8 Universal character set. Note that you cannot change the character set after the DB instance is created.
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For any non-trivial instance, you should set this value to **`1`** or greater.
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of **`No Preference`**.

For this parameter...	...Do this:
Auto Minor Version Upgrade	Select **`Yes`** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Maintenance Window	Select the 30 minute window in which pending modifications to your DB instance are applied. If you the time period doesn't matter, select **`No Preference`**.

9. On the final page of the wizard, click **Close**.
10. On the RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and storage allocated, it could take several minutes for the new instance to be available.



CLI

To create an Oracle DB instance

- Use the command `rds-create-db-instance` to create a DB instance. The following command will launch the example DB instance.

```
PROMPT>rds-create-db-instance mydbinstance -s 20 -c db.m1.small -e oracle-
se1
   - u <masterawsuser> -p <masteruserpassword> --backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE  mydbinstance  db.m1.small  oracle-se1  20  sa  creating  3  ****
  n  11.2.0.3.v1
SECGROUP  default  active
PARAMGRP  default.oracle-se1-11.2  in-sync
```

API

To create an Oracle DB instance

- Call the `CreateDBInstance` action. For example, you could use the following parameters:
  - *`DBInstanceIdentifier`* = `mydbinstance`
  - *`Engine`* = `oracle-se1`
  - *`DBInstanceClass`* = `db.m1.small`

- *AllocatedStorage* = 20
- *BackupRetentionPeriod* = 3
- *MasterUsername* = <masterawsuser>
- *MasterUserPassword* = <masteruserpassword>

Example

```
https://rds.amazonaws.com/
   ?Action=CreateDBInstance
   &AllocatedStorage=20
   &BackupRetentionPeriod=3
   &DBInstanceClass=db.m1.small
   &DBInstanceIdentifier=mydbinstance
   &DBName=mydatabase
   &DBSecurityGroups.member.1=mysecuritygroup
   &DBSubnetGroup=mydbsubnetgroup
   &Engine=oracle-se1
   &MasterUserPassword=<masteruserpassword>
   &MasterUsername=<masterawsuser>
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140202/us-west-2/rds/aws4_request
   &X-Amz-Date=20140202T190545Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=60e907d8d43fdc978941c1566f7b3c5054e0328622a871fb59b61782ee1f30d8
```

Related Topics

- Amazon RDS DB Instances (p. 64)
- Amazon RDS Security Groups (p. 110)
- DB Instance Class (p. 65)
- Deleting a DB Instance (p. 459)

Connecting to a DB Instance Running the Oracle Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. In this example, you connect to a DB instance running the Oracle database engine using the Oracle command line tools. For more information on using Oracle, go to the Oracle website.

> **Note**
> This example uses the Oracle *sqlplus* command line utility. This utility is part of the Oracle software distribution. To download a stand-alone version of this utility, go to the SQL*Plus User's Guide and Reference.

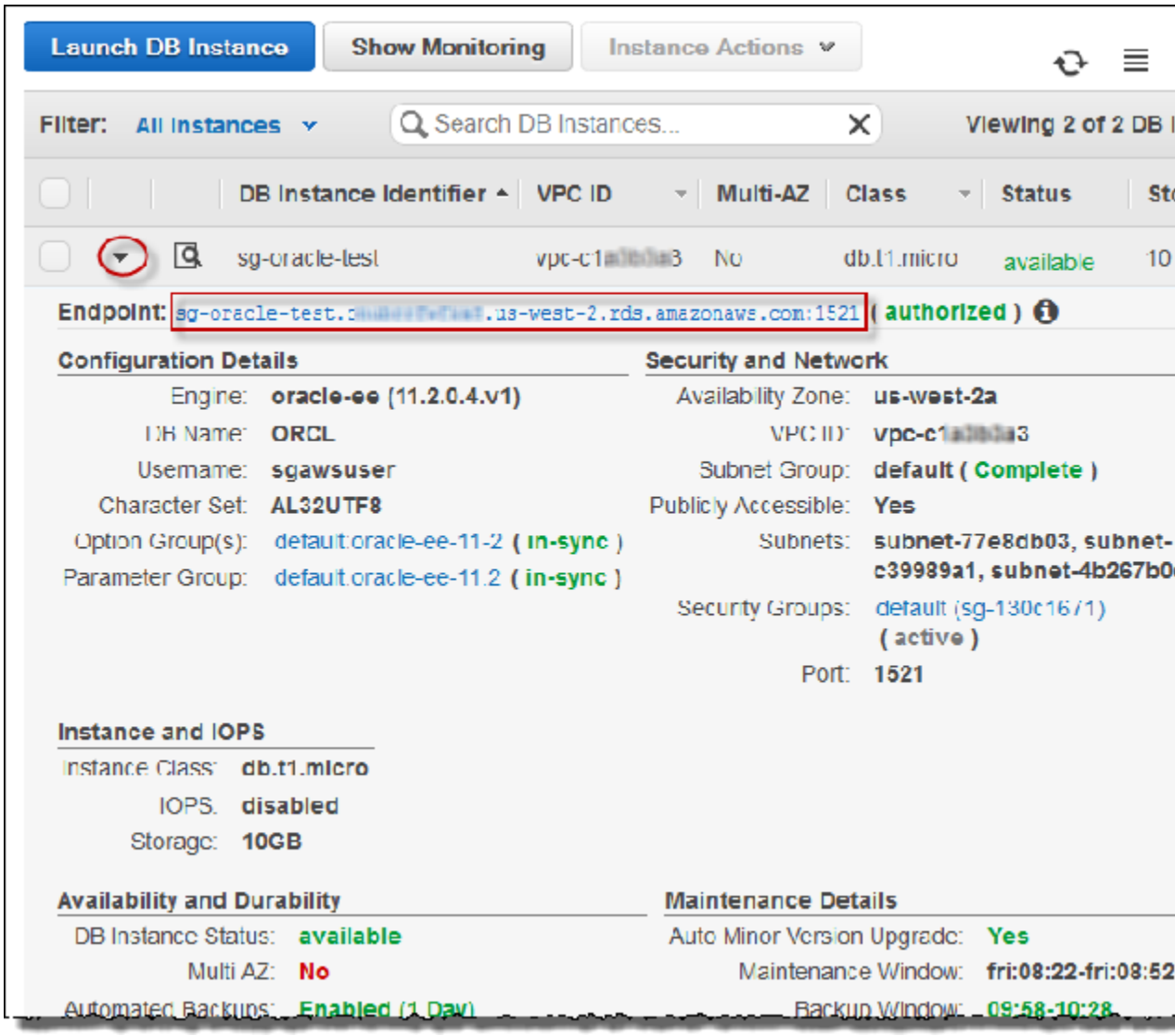
Console

To connect to an Oracle DB instance with Information from the RDS Console

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. In this example, you connect to a DB instance running the Oracle database engine using the Oracle command line tools. For more information on using Oracle, go to the Oracle website.

This example uses the Oracle *sqlplus* command line utility. This utility is part of the Oracle software distribution. To download a stand-alone version of this utility, go to the SQL*Plus User's Guide and Reference.

1. Open the RDS console, then select **Instances** in the left column to display a list of your DB instances.
2. In the row for your Oracle DB instance, select the arrow to display the summary information for the instance.
3. The **Endpoint** field contains part of the connection information for your DB instance. The **Endpoint** field has two parts separated by a colon (:). The part before the colon is the DNS name for the instance, the part following the colon is the port.



4. Type the following command on one line at a command prompt to connect to a DB instance using the sqlplus utility. The value for `Host` will be the DNS name for your DB instance, the value for `Port` will be the port you assigned the DB instance, and the value for the Oracle `SID` will be the name of the DB instance's database that you specified when you created the DB instance, not the name of the DB instance.

```
PROMPT>sqlplus 'mydbusr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<endpoint>)
    (PORT=<port number>))(CONNECT_DATA=(SID=<database name>)))'
```

You will see output similar to the following.

```
SQL*Plus: Release 11.1.0.7.0 - Production on Wed May 25 15:13:59 2011

SQL>
```

CLI

To connect to a DB Instance using sqlplus

1. Find the DNS name for your DB instance using the rds-describe-db-instances command below, or use the Amazon RDS console to find the necessary connection information.

```
PROMPT>rds-describe-db-instances --headers
```

   You will see output similar to the following:

```
DBINSTANCE  DBInstanceId   Created                  Class        Engine
Storage
  Master Username  Status     Endpoint Address
       Port  AZ          Backup Retention  Multi-AZ  Version  Read Replica
 Source
 ID  License
DBINSTANCE  oracledb   2011-05-14T01:11:01.727Z  db.m1.small  oracle-ee
20
   mydbusr          available  oracledb.mydnsnameexample.rds.amazonaws
.com   1521  us-east-1a  1                n         11.2.0.2.v3
      bring-your-own-license
```

2. Type the following command on one line at a command prompt to connect to a DB instance using the *sqlplus* utility. Substitute the DNS name for your DB instance, then include the port and the Oracle SID. The SID value is the name of the instance's database that you specified when you created the DB instance, not the name of the DB instance. When using *sqlplus* from a Windows command line, do not use the single quotes.

```
PROMPT>sqlplus 'mydbusr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<dns name
 of db instance>)
    (PORT=<listener port>))(CONNECT_DATA=(SID=<database name>)))'
```

   Note
   The shorter format connection string, such as `PROMPT>sqlplus USER/PASSWORD@LONGER-THAN-63-CHARS-RDS-ENDPOINT-HERE:1521/DATABASE_IDENTIFIER`, may encounter a maximum character limit and should not be used to connect.

   You will see output similar to the following.

```
SQL*Plus: Release 11.1.0.7.0 - Production on Wed May 25 15:13:59 2011

SQL>
```

Related Topics

- Amazon RDS DB Instances (p. 64)

- Creating a DB Instance Running the MySQL Database Engine (p. 129)
- Amazon RDS Security Groups (p. 110)
- Deleting a DB Instance (p. 459)

Modifying a DB Instance Running the Oracle Database Engine

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. This topic guides you through modifying an Amazon RDS Oracle DB instance, and describes the settings for Oracle instances. For information about additional tasks, such as renaming, rebooting, deleting, tagging, or upgrading an Amazon RDS DB instance, see Amazon RDS DB Instance Lifecycle (p. 440).

Before you upgrade your production DB instances to a new Oracle Database version, we recommend you test the upgrade process on a test instance to verify its duration and to validate your applications. We do not recommend upgrading micro DB instances because they have limited CPU resources and the upgrade process may take hours to complete. An alternative to upgrading micro DB instances with small storage (10-20 GB) would be to copy your data using Data Pump, where we also recommend testing before migrating your production instances.

You can have the changes apply immediately or have them applied during the DB instance's next maintenance window. Applying changes immediately can cause an outage in some cases; for more information on the impact of the **Apply Immediately** option when modifying a DB instance, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 455).

AWS Management Console

To modify an Oracle DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **DB Instances**.
3. Select the check box for the DB instance that you want to change, and then click **Modify**.
4. In the **Modify DB Instance** dialog box, change any of the following settings that you want:

Setting	Description
DB Engine Version	In the list provided, click the version of the Oracle database engine that you want to use. Before you upgrade your production database instances, we recommend you test the upgrade process on a test instance to verify its duration and to validate your applications. We do not recommend upgrading micro DB instances because they have limited CPU resources and the upgrade process may take hours to complete. An alternative to upgrade micro DB instances with small storage (10-20 GB) would be to copy your data using Data Pump, where we also recommend testing before migrating your production instances.
DB Instance Class	In the list provided, click the DB instance class that you want to use. For information about instance classes, see DB Instance Class (p. 65).
Multi-AZ Deployment	If you want to deploy your DB instance in multiple Availability Zones, click **Yes**; otherwise, click **No**.
Allocated Storage	Specify how much storage, in gigabytes, will be initially allocated for your DB instance. The minimum allowable value is 10 GB; the maximum is 6 TB.

Setting	Description
Storage Type	Select the storage type you want to use. Changing from **Magnetic** to **General Purpose (SSD)** or **Provisioned IOPS (SSD)** will result in an outage. Also, changing from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic** will result in an outage. For more information about storage, see Storage for Amazon RDS (p. 77).
DB Instance Identifier	You can rename the DB instance by typing a new name. When you change the DB instance identifier, an instance reboot will occur immediately if you set `Apply Immediately` to true, or will occur during the next maintenance window if you set `Apply Immediately` to false. This value is stored as a lowercase string.
New Master Password	Type a password for your master user. The password must contain from 8 to 30 alphanumeric characters.
Security Group	Select the security group you want associated with the DB instance. For more information about security groups, see Working with DB Security Groups (p. 537).
Parameter Group	Select the parameter group you want associated with the DB instance. Changing this setting does not result in an outage. The parameter group name itself is changed immediately, but the actual parameter changes are not applied until you reboot the instance without failover. The DB instance will NOT be rebooted automatically and the parameter changes will NOT be applied during the next maintenance window. For more information about parameter groups, see Working with DB Parameter Groups (p. 523).
Option Group	Select the option group you want associated with the DB instance. For more information about option groups, see Working with Option Groups (p. 510).
Backup Retention Period	Specify the number of days that automatic backups will be retained. To disable automatic backups, set this value to 0. **Note** An immediate outage will occur if you change the backup retention period from 0 to a non-zero value or from a non-zero value to 0.
Backup Window	Set the time range during which automated backups of your databases will occur. Specify a start time in Universal Coordinated Time (UTC) and a duration in hours.
Auto Minor Version Upgrade	If you want your DB instance to receive minor engine version upgrades automatically when they become available, click **Yes**. Upgrades are installed only during your scheduled maintenance window.
Maintenance Window	Set the time range during which system maintenance, including upgrades, will occur. Specify a start time in UTC and a duration in hours.

5. To apply the changes immediately, select the **Apply Immediately** check box. Selecting this option can cause an outage in some cases; for more information on the impact of the **Apply Immediately** option, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 455).
6. When all the changes are as you want them, click **Yes, Modify**. If instead you want to cancel any changes that you didn't apply in the previous step, click **Cancel**.

CLI

To modify an Oracle DB instance

- Use the command `rds-modify-db-instance`.

API

To modify an Oracle DB instance

- Use the `ModifyDBInstance` `action`.

Importing Data Into Oracle on Amazon RDS

How you import data into an Amazon RDS DB instance depends on the amount of data you have and the number and variety of database objects in your database. For example, you can use Oracle SQL Developer to import a simple, 20 MB database; you want to use Oracle Data Pump to import complex databases or databases that are several hundred megabytes or several terabytes in size.

Before you use any of these migration techniques, we recommend the best practice of taking a backup of your database. You can back up your Amazon RDS instances by creating snapshots. Later, you can restore the database from the snapshots using the Restore from DB Snapshot or Restore to Point In Time options on the RDS tab of the AWS Management Console. You can also use the Amazon RDS command line methods `rds-restore-db-instance-from-db-snapshot` or `rds-restore-db-instance-to-point-in-time`. These and other best practices are addressed in this section.

Oracle SQL Developer

For small databases, you can use Oracle SQL Developer, a graphical Java tool distributed without cost by Oracle. You can install this tool on your desktop computer (Windows, Linux, or Mac) or on one of your servers. Oracle SQL Developer provides options for migrating data between two Oracle databases, or for migrating data from other databases, such as MySQL, to Oracle. Oracle SQL Developer is best suited for migrating small databases. We recommend that you read the Oracle SQL Developer product documentation before you begin migrating your data.

After you install SQL Developer, you can use it to connect to your source and target databases. Use the Database Copy command on the Tools menu to copy your data to your Amazon RDS instance.

To download Oracle SQL Developer, go to http://www.oracle.com/technetwork/developer-tools/sql-developer.

Oracle also has documentation on how to migrate from other databases, including MySQL and SQL Server. To learn more, go to http://www.oracle.com/technetwork/database/migration.

Oracle Data Pump

Oracle Data Pump is a long-term replacement for the Oracle Export/Import utilities and is the preferred way to move large amounts of data from an Oracle installation to an Amazon RDS DB instance. You can use Oracle Data Pump for several scenarios:

- Import data from an Amazon EC2 instance with an Oracle database to an Oracle DB instance
- Import data from a database on an Oracle DB instance to another Oracle DB instance
- Import data from a database on an Oracle DB instance in a VPC to another Oracle DB instance with or without a VPC
- Import data from a local Oracle database to an Amazon RDS DB instance

The following process uses Oracle Data Pump and the DBMS_FILE_TRANSFER package. The process connects to an Oracle instance and exports data using Oracle Data Pump. It then uses the DBMS_FILE_TRANSFER.PUT_FILE method to copy the dump file from the Oracle instance to the DATA_PUMP_DIR on the target DB instance that is connected via a database link. The final step imports the data from the copied dump file into the RDS instance.

The process has the following requirements:

- You must have execute privileges on the DBMS_FILE_TRANSFER package
- The target DB instance must be version 11.2.0.2.v6 or later

- You must have write privileges to the DATA_PUMP_DIR directory on the source DB instance
- You must ensure that you have enough storage space to store the dump file on the source instance and the target DB instance

Note
This process imports a dump file into the DATA_PUMP_DIR directory, a preconfigured directory on all Oracle DB instances. This directory is located on the same storage volume as your data files. When you import the dump file, the existing Oracle data files will use more space, so you should make sure that your DB instance can accommodate that additional use of space as well. Given that the current storage size limit of an Oracle DB instance is 3 TB, you can't use this process to upload any dump file larger than about 1.3 TB. Note that the imported dump file is not automatically deleted or purged from the DATA_PUMP_DIR directory. Use UTL_FILE.FREMOVE to remove the imported dump file.

The import process using Oracle Data Pump and the DBMS_FILE_TRANSFER package has the following steps:

- Step 1: Grant privileges to user on source database
- Step 2: Use DBMS_DATAPUMP to create a dump file
- Step 3: Create a database link to the target DB instance
- Step 4: Use DBMS_FILE_TRANSFER to copy the exported dump file to the Amazon RDS instance
- Step 5: Import the dump file into a database on the Amazon RDS instance
- Step 6: Clean up

Step 1: Grant privileges to user on source database

Use SQL Plus or Oracle SQL Developer to connect to the Oracle instance that contains the data to be imported. If necessary, create a user account and grant the necessary permissions.

The following commands create a new user and grant the necessary permissions:

```
SQL> create user USER1 identified by test123;
SQL> grant create session, create table to USER1;
SQL> alter user USER1 quota 100M on users;
SQL> grant read, write on directory data_pump_dir to USER1;
SQL> grant execute on dbms_datapump to USER1;
```

You can use your own table, or you can create one to test the process. The following commands create a sample table for importing into a DB instance:

```
SQL> create table USER1.tab1
tablespace users
as select 'USER1_'||object_name str_col, sysdate dt_col from all_objects;
```

Step 2: Use DBMS_DATAPUMP to create a dump file

Use SQL Plus or Oracle SQL Developer to connect to the Oracle instance and use the Oracle Data Pump utility to create a dump file. The following script creates a dump file named *tab1.dmp* in the DATA_PUMP_DIR directory.

```
DECLARE
hdnl NUMBER;
BEGIN
hdnl := DBMS_DATAPUMP.open( operation => 'EXPORT', job_mode => 'SCHEMA',
job_name=>null);
DBMS_DATAPUMP.ADD_FILE( handle => hdnl, filename => 'tab1.dmp', directory =>
'DATA_PUMP_DIR', filetype => dbms_datapump.ku$_file_type_dump_file);
DBMS_DATAPUMP.add_file( handle => hdnl, filename => 'exp.log', directory =>
'DATA_PUMP_DIR', filetype => dbms_datapump.ku$_file_type_log_file);
DBMS_DATAPUMP.METADATA_FILTER(hdnl,'SCHEMA_EXPR','IN (''USER1'')');
DBMS_DATAPUMP.start_job(hdnl);
END;
/
```

Step 3: Create a database link to the target DB instance

Next, create a database link between your source instance and your target DB instance. Note that your local Oracle instance must have network connectivity to the DB instance in order to create a database link and to transfer your export file.

The following command creates a database link named *to_rds* to another user at the target DB instance database:

> **Note**
> If you are creating a database link between two DB instances inside a VPC, the two DB instances must be either in the same VPC, be in VPCs that have an established VPC peering connection, or you must create an EC2 or VPC security group that both DB instances are a member of.

```
create database link to_rds connect to USER2 identified by user2pwd
using '(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST="<dns or ip address of remote
 db>)(PORT=<listener port>))(CONNECT_DATA=(SID=<remoteSID>)))';
```

Step 4: Use DBMS_FILE_TRANSFER to copy the exported dump file to an Amazon RDS DB instance

Next, use DBMS_FILE_TRANSFER to copy the dump file from the source database instance to the target DB instance. The following script copies a dump file named tab1.dmp from the source instance to a target database link named *to_rds* (created in the previous step):

```
BEGIN
  DBMS_FILE_TRANSFER.PUT_FILE(
```

```
    source_directory_object       => 'DATA_PUMP_DIR',
    source_file_name              => 'tab1.dmp',
    destination_directory_object  => 'DATA_PUMP_DIR',
    destination_file_name         => 'tab1_copied.dmp',
    destination_database          => 'to_rds'
  );
END;
/
```

Step 5: Create the Necessary Tablespace on the Target Instance

You must create the tablespace before you can import the data. See the topic Creating and Resizing Tablespaces and Data Files (p. 249) for more information about creating tablespaces.

Step 6: Use Data Pump to import the data file on the DB instance

Use Oracle Data Pump to import the schema in the DB instance. The first part of the listing shows the format for the data import statement, and the second part shows importing a data file called *tab1_copied.dmp*. Note that additional options such as REMAP_TABLESPACE might be required.

```
impdp <username>@<TNS_ENTRY> DUMPFILE=user1copied.dmp DIRECTORY=DATA_PUMP_DIR
full=y

impdp copy1@copy1 DUMPFILE=tab1_copied.dmp DIRECTORY=DATA_PUMP_DIR full=y
```

You can verify the data import by viewing the table on the DB instance.

```
SQL> select count(*) from user1.tab1;
```

Step 7: Clean up

After the data has been imported, you can delete the files you no longer want to keep. You can list the files in the DATA_PUMP_DIR using the following command:

```
select * from table(RDSADMIN.RDS_FILE_UTIL.LISTDIR('DATA_PUMP_DIR')) order by
mtime;
```

The following command can be used to delete files in the DATA_PUMP_DIR that you no longer require:

```
exec utl_file.fremove('DATA_PUMP_DIR','[file name]');
```

For example, the following command deletes the file named "test_dbms_lob.txt" :

```
exec utl_file.fremove('DATA_PUMP_DIR','test_dbms_lob.txt');
```

Oracle Export/Import Utilities

The Oracle Export/Import utilities are best suited for migrations where the data size is small and data types such as binary float and double are not required. The import process creates the schema objects so you do not need to run a script to create them beforehand, making this process well suited for databases with small tables. The following example demonstrates how these utilities can be used to export and import specific tables.

Export the tables from the source database using the command below. Substitute username/password as appropriate.

```
exp cust_dba@ORCL FILE=exp_file.dmp TABLES=(tab1,tab2,tab3) LOG=exp_file.log
```

The export process creates a binary dump file that contains both the schema and data for the specified tables. Now this schema and data can be imported into a target database using the command:

```
imp cust_dba@targetdb FROMUSER=cust_schema TOUSER=cust_schema \
TABLES=(tab1,tab2,tab3) FILE=exp_file.dmp LOG=imp_file.log
```

There are other variations of the Export and Import commands that might be better suited to your needs. See Oracle's documentation for full details.

Oracle SQL*Loader

Oracle SQL*Loader is well suited for large databases that have a limited number of objects in them. Since the process involved in exporting from a source database and loading to a target database is very specific to the schema, the following example creates the sample schema objects, exports from a source, and then loads it into a target database.

1. Create a sample source table using the command below.

```
create table customer_0 tablespace users as select rownum id, o.* from
all_objects o, all_objects x where rownum <= 1000000;
```

2. On the target Amazon RDS instance, create a destination table that will be used to load the data.

```
create table customer_1 tablespace users as select 0 as id, owner,
object_name, created from all_objects where 1=2;
```

3. The data will be exported from the source database to a flat file with delimiters. This example uses SQL*Plus for this purpose. For your data, you will likely need to generate a script that does the export for all the objects in the database.

```
alter session set nls_date_format = 'YYYY/MM/DD HH24:MI:SS'; set linesize 800

HEADING OFF FEEDBACK OFF array 5000 pagesize 0 spool customer_0.out SET
MARKUP HTML PREFORMAT ON SET COLSEP ',' SELECT id, owner, object_name,
created FROM customer_0; spool off
```

4. You need to create a control file to describe the data. Again, depending on your data, you will need to build a script that does this step.

```
cat << EOF > sqlldr_1.ctl
load data
infile customer_0.out
into table customer_1
APPEND
fields terminated by "," optionally enclosed by '"'
(
id              POSITION(01:10)         INTEGER EXTERNAL,
owner             POSITION(12:41)         CHAR,
object_name       POSITION(43:72)         CHAR,
created           POSITION(74:92)          date "YYYY/MM/DD HH24:MI:SS"
)
```

   If needed, copy the files generated by the preceding code to a staging area, such as an Amazon EC2 instance.

5. Finally, import the data using SQL*Loader with the appropriate username and password for the target database.

```
sqlldr cust_dba@targetdb control=sqlldr_1.ctl BINDSIZE=10485760 READ
SIZE=10485760 ROWS=1000
```

Oracle Materialized Views

You can also make use of Oracle materialized view replication to migrate large datasets efficiently. Replication allows you to keep the target tables in sync with the source on an ongoing basis, so the actual cutover to Amazon RDS can be done later, if needed. The replication is set up using a database link from the Amazon RDS instance to the source database.

One requirement for materialized views is to allow access from the target database to the source database. In the following example, access rules were enabled on the source database to allow the Amazon RDS target database to connect to the source over SQLNet.

1. Create a user account on both source and Amazon RDS target instances that can authenticate with the same password.

```
create user dblink_user identified by password      default tablespace users

temporary tablespace temp; grant create session to dblink_user; grant select

any table to dblink_user; grant select any dictionary to dblink_user;
```

2. Create a database link from the Amazon RDS target instance to the source instance using the newly created dblink_user.

```
create database link remote_site
  connect to dblink_user identified by password
  using '(description=(address=(protocol=tcp) (host=<myhost>) (port=<listener
 port>))
(connect_data=(sid=<sourcedb sid>)))';
```

3. Test the link:

```
select * from v$instance@remote_site;
```

4. Create a sample table with primary key and materialized view log on the source instance.

```
create table customer_0 tablespace users as select rownum id, o.* from
all_objects o, all_objects x where rownum <= 1000000; alter table customer_0

add constraint pk_customer_0 primary key (id) using index; create
materialized view log on customer_0;
```

5. On the target Amazon RDS instance, create a materialized view.

```
CREATE MATERIALIZED VIEW customer_0    BUILD IMMEDIATE   REFRESH FAST    AS
SELECT * FROM cust_dba.customer_0@remote_site;
```

Appendix: Options for Oracle Database Engine

This appendix describes options, or additional features, that are available for Amazon RDS instances running the Oracle database engine. To enable these options, you can add them to an option group, and then associate the option group with your DB instance. Note that some options are permanent and persistent; permanent means that an option cannot be removed from an option group and persistent means that once an option group with this option is assigned to a DB instance, the option group cannot be removed from the DB instance. For more information about working with options, see Option Groups Overview (p. 510).

The following options are currently supported for Oracle:

- Oracle 11g Enterprise Manager (OEM) Database Control and Oracle 12c OEM Database Express (p. 226)
- Oracle XML DB (p. 227)
- Oracle Application Express (APEX) (p. 227)
- Oracle Native Network Encryption (p. 233)
- Oracle Transparent Data Encryption (TDE) (p. 235) (a feature of the Oracle Advanced Security option available in Oracle Enterprise Edition)
- Oracle Statspack (p. 236)
- Oracle Time Zone (p. 239)

Note
Some of these options may require additional memory in order to run on your DB instance. For example, Oracle Enterprise Manager Database Control uses about 300 MB of RAM; if you enable this option for a small DB instance, you might encounter performance problems due to memory constraints.
Before you enable these options, please consider whether your DB instance has enough available memory. You can adjust the Oracle parameters so that the database requires less RAM; alternatively, you can scale up to a larger DB instance.

Oracle 11g Enterprise Manager (OEM) Database Control and Oracle 12c OEM Database Express

Oracle Enterprise Manager (OEM) Database Control for Oracle version 11g and Oracle Enterprise Manager (OEM) Database Express for Oracle version 12c are similar tool that have a web-based interface for Oracle database administration. Note that neither tool can be run on DB instances that use the db.t1.micro or db.m1.small instance classes.

The default port number for OEM Database Control is 1158; the default port number for OEM Database Express is 5500. You can either accept the port number or choose a different one when you enable the option for your DB instance. You can then go to your web browser and begin using the OEM tool for your Oracle version.

The following example shows how to access either OEM Database Control or OEM Database Express from your web browser. Suppose that the endpoint for your Amazon RDS instance is *mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com*, and that you specified port 1158. The URL to access OEM Database Control would be:

```
https://mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com:1158/em
```

In this example, the OEM Database Control login window appears, prompting you for a username and password. Enter the master username and master password for your DB instance. You are now ready to manage your Oracle databases.

Oracle XML DB

Oracle XML DB adds native XML support to your DB instance. With the Amazon RDS XMLDB option, DB instances running the Oracle engine can store and retrieve structured or unstructured XML, in addition to relational data.

After you apply the XMLDB option to your DB instance, you have full access to the Oracle XML DB repository; no post-installation tasks are required.

> **Note**
> The Amazon RDS XMLDB option does not provide support for the Oracle XML DB Protocol Server.

Oracle Application Express (APEX)

Oracle Application Express (APEX) is a development and runtime environment for web-based applications. Using APEX, developers can build applications entirely within the web browser, and customers can run these applications without installing any additional software.

> **Note**
> Amazon RDS Oracle 11g supports Oracle APEX version 4.1.1. Amazon RDS 12c supports Oracle APEX version 4.2.0.00.08 and later.

Topics

- Oracle APEX on Amazon RDS Oracle 11g (p. 227)
- Oracle APEX on Amazon RDS Oracle 12c (p. 230)

Oracle APEX consists of two main components:

- A *repository* that stores the metadata for APEX applications and components. The repository consists of tables, indexes, and other objects that are installed in your Amazon RDS DB instance.
- A *listener* that manages HTTP communications with APEX clients. The listener accepts incoming connections from web browsers and forwards them to the Amazon RDS instance for processing, and then sends results from the repository back to the browsers. The APEX Listener was renamed Oracle Rest Data Services (ORDS) in Oracle 12c.

When you add the APEX option for your Oracle DB instance, Amazon RDS installs the APEX repository only. You must install the listener on a separate host — an Amazon EC2 instance, an on-premises server at your company, or your desktop computer.

The following sections explain how to configure the Oracle APEX repository and listener for use with Amazon RDS.

Oracle APEX on Amazon RDS Oracle 11g

The setup of Oracle APEX for Oracle 11g DB instances requires that you install the XMLDB option as well as the APEX and APEX_DEV options on the repository.

Repository Configuration for Oracle 11g

To configure the APEX repository for Oracle 11g

1. Create a new Amazon RDS instance running the Oracle engine, or choose an existing instance. The version number for the Oracle engine must be 11.2.0.2.v4 or newer.
2. Create a new option group, or select an existing option group. Apply the following options to this option group:

   - XMLDB
   - APEX
   - APEX_DEV

   (If you only want to deploy the APEX runtime environment, you can remove the APEX_DEV option at a later time. This option must be present during this configuration procedure, however.)
3. Apply the option group to your DB instance. Amazon RDS will install the repository components in your DB instance; this process takes a few minutes to complete.
4. After the option group is successfully applied, you will need to change the password for the APEX_PUBLIC_USER database account and unlock it. You can do this using the Oracle SQL*Plus command line utility: Connect to your DB instance as the master user and issue the following commands:

   ```
   alter user APEX_PUBLIC_USER identified by newpass;
   alter user APEX_PUBLIC_USER account unlock;
   ```

   Replace `newpass` with a password of your choice.

Listener Configuration for Oracle 11g

You are now ready to configure a listener for use with Oracle APEX. You can use either of these products for this purpose:

- Oracle Application Express Listener
- Oracle HTTP Server and *mod_plsql*

Note
Amazon RDS does not support the Oracle XML DB HTTP server with the embedded PL/SQL gateway; you cannot use this as an APEX listener. This restriction is in line with Oracle's recommendation against using the embedded PL/SQL gateway for applications that run on the Internet.

The listener must be installed on a separate host, such as an Amazon EC2 instance or a server that you own. You also must have the following prerequisite software installed on the separate host acting as the listener:

- Java Runtime Environment (JRE) — Oracle APEX Listener is a Java application.
- Oracle Net Services, to enable the APEX listener to connect to your Amazon RDS instance.
- SQL*Plus, to perform administrative tasks from the command line.

The following procedure shows how to configure the Oracle Application Express Listener product. We will assume that the name of your APEX host is *myapexhost.example.com*, and that this host is running Linux.

To configure an APEX listener for Oracle 11g

1. Log in to *myapexhost.example.com* as *root*.
2. We recommend that you create a nonprivileged OS user to own the APEX listener installation. The following command will create a new user named *apexuser*:

   ```
   useradd -d /home/apexuser apexuser
   ```

   Now assign a password to *apexuser*:

   ```
   passwd apexuser
   ```

3. Log in to *myapexhost.example.com* as *apexuser*, and download the APEX and APEX Listener installation files from Oracle:

   - http://www.oracle.com/technetwork/developer-tools/apex/downloads/index.html
   - http://www.oracle.com/technetwork/developer-tools/apex-listener/downloads/index.html

4. Open the APEX file:

   ```
   unzip apex_4.1.1.zip
   ```

5. Create a new directory and open the APEX Listener file:

   ```
   mkdir /home/apexuser/apexlistener
   cd /home/apexuser/apexlistener
   unzip ../apex_listener.1.1.3.243.11.40.zip
   ```

6. While you are still in the *apexlistener* directory, run the APEX Listener program:

   ```
   java -Dapex.home=./apex -Dapex.images=/home/apexuser/apex/images -Dapex.erase
    -jar ./apex.war
   ```

   The program will prompt you for the following:

   - The APEX Listener Administrator username — the default is *adminlistener*
   - A password for the APEX Listener Administrator.
   - The APEX Listener Manager username — the default is *managerlistener*
   - A password for the APEX Listener Administrator.

The program will print a URL that you will need in order to complete the configuration:

```
INFO: Please complete configuration at: http://localhost:8080/apex/listener
Configure
Database is not yet configured
```

Leave the APEX Listener running. It needs to continue running in order for you to use Oracle Application Express. (When you have finished this configuration procedure, you can run the listener in the background.)

7. From your web browser, go to the URL provided by the APEX Listener program. The Oracle Application Express Listener administration window appears. Enter the following information:

   - **Username**— *APEX_PUBLIC_USER*
   - **Password** — the password for *APEX_PUBLIC_USER*. (This is the password that you specified earlier, when you configured the APEX repository.)
   - **Connection Type**— Basic
   - **Hostname**— the endpoint of your Amazon RDS instance, such as *mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com*
   - **Port**— 1521
   - **SID**— the name of the database on your Amazon RDS instance, such as *mydb*

   Click **Apply** button. The APEX administration window appears.
8. You will need to set a password for the APEX *admin* user. To do this, use SQL*Plus to connect to your DB instance as the master user and issue the following commands:

   ```
   grant APEX_ADMINISTRATOR_ROLE to master;
   @/home/apexuser/apex/apxchpwd.sql
   ```

   Replace `master` with your master user name. Enter a new *admin* password when the *apxchpwd.sql* script prompts you.
9. Return to the APEX administration window in your browser and click **Administration**. Next, click **Application Express Internal Administration**. You will be prompted for APEX internal administration credentials. Enter the following information:

   - **Username**— *admin*
   - **Password**— the password you set using the *apxchpwd.sql* script.

   Click **Login**. You will be required to set a new password for the *admin* user.

Oracle Application Express is now ready for use.

Oracle APEX on Amazon RDS Oracle 12c

The installation process for installing the repository for Oracle 12c is the same as Oracle 11g except that you no longer have to install the XMLDB option (it is installed by default in Oracle 12c.). Note that the APEX Listener was renamed Oracle Rest Data Services (ORDS).

Repository Configuration for Oracle 12c

To configure the APEX repository for Oracle 12c

1. Create a new Amazon RDS instance running the Oracle 12c DB engine, or choose an existing Oracle 12c DB instance.
2. Create a new option group, or select an existing option group. Apply the following options to this option group:

   - APEX
   - APEX_DEV

   (If you only want to deploy the APEX runtime environment, you can remove the APEX_DEV option at a later time. This option must be present during this configuration procedure, however.)
3. Apply the option group to your DB instance. Amazon RDS will install the repository components in your DB instance; this process takes a few minutes to complete.
4. After the option group is successfully applied, you will need to change the password for the APEX_PUBLIC_USER database account and unlock it. You can do this using the Oracle SQL*Plus command line utility: Connect to your DB instance as the master user and issue the following commands:

   ```
   alter user APEX_PUBLIC_USER identified by newpass;
   alter user APEX_PUBLIC_USER account unlock;
   ```

   Replace `newpass` with a password of your choice.

Listener Configuration for Oracle 12c

For Oracle 12c, the Oracle Application Express Listener (APEX Listener) was renamed Oracle Rest Data Services (ORDS). The listener must be installed on a separate host, such as an Amazon EC2 instance or a server that you own.

Amazon RDS does not support the Oracle XML DB HTTP server with the embedded PL/SQL gateway; you cannot use this as an APEX Listener. This restriction is in line with Oracle's recommendation against using the embedded PL/SQL gateway for applications that run on the Internet.

You must have the following prerequisite software installed on the separate host acting as the listener:

- Java Runtime Environment (JRE)
- Oracle Net Services, to enable the APEX Listener to connect to your Amazon RDS instance.
- SQL*Plus, to perform administrative tasks from the command line.

The following procedure shows how to configure Oracle Rest Data Services (ORDS). We will assume that the name of your APEX host is *myapexhost.example.com*, and that this host is running Linux.

To install Oracle Rest Data Services (ORDS) (the APEX listener) for Oracle 12c

1. Log in to *myapexhost.example.com* as *root*.
2. We recommend that you create a nonprivileged OS user to own the APEX listener installation. The following command will create a new user named *apexuser*:

```
useradd -d /home/apexuser apexuser
```

Now assign a password to *apexuser*:

```
passwd apexuser
```

3. Log in to *myapexhost.example.com* as *apexuser*, and download the APEX and ORDS installation files from Oracle:

   - http://www.oracle.com/technetwork/developer-tools/apex/downloads/index.html
   - http://www.oracle.com/technetwork/developer-tools/apex-listener/downloads/index.html

4. Unzip the APEX file:

```
unzip apex_4.2.6.zip
```

5. Create a new directory and open the ORDS file:

```
mkdir /home/apexuser/ORDS
cd /home/apexuser/ORDS
unzip ../ords.3.0.0.65.09.31.zip
```

6. While you are still in the *ORDS* directory, run the APEX Listener program:

```
java -jar ords.war setup
```

The program will prompt you for the following information. The default value is in brackets:

- Enter the name of the database server [localhost]:
- Enter the database listen port [1521]:
- Enter 1 to specify the database service name, or 2 to specify the database SID [1]:
- Enter the database SID [xe]:
- Enter the database user name [APEX_PUBLIC_USER]:
- Enter the database password:

7. You will need to set a password for the APEX *admin* user. To do this, use SQL*Plus to connect to your DB instance as the master user and issue the following commands:

```
grant APEX_ADMINISTRATOR_ROLE to master;
@/home/apexuser/apex/apxchpwd.sql
```

Replace `master` with your master user name. Enter a new *admin* password when the *apxchpwd.sql* script prompts you.

8. Start the APEX Listener.

```
java -jar ords.war
```

The first time you start the APEX Listener, you will be prompted to provide the location of the APEX Static resources. This images folder is located under the installation directory for APEX, then /apex/images.

9. Return to the APEX administration window in your browser and click **Administration**. Next, click **Application Express Internal Administration**. You will be prompted for APEX internal administration credentials. Enter the following information:

   - **Username**— *admin*
   - **Password**— the password you set using the *apxchpwd.sql* script.

   Click **Login**. You will be required to set a new password for the *admin* user.

Oracle Application Express (APEX) is now ready for use.

Oracle Native Network Encryption

Amazon RDS supports Oracle native network encryption, a feature available on all Oracle versions. With native network encryption, you can encrypt data as it moves to and from a DB instance.

To use Oracle native network encryption with a DB instance, you add the NATIVE_NETWORK_ENCRYPTION option to an option group and associate that option group with the DB instance. You should first determine if the DB instance is associated with an option group that has the NATIVE_NETWORK_ENCRYPTION option. To view the option group that a DB instance is associated, you can use the RDS console, the rds-describe-db-instance CLI command, or the API action DescribeDBInstances. Amazon RDS supports Oracle native network encryption for any DB instance class larger than db.t1.micro.

A detailed discussion of Oracle native network encryption is beyond the scope of this guide, but you should understand the strengths and weaknesses of each algorithm and key before you decide on a solution for your deployment. Note that non-default TDE encryption algorithms only work with Oracle version 11.2.0.2.v7 and later. For information about the algorithms and keys that are available through Oracle native network encryption, see Configuring Network Data Encryption in the Oracle documentation. For more information about AWS security, see the AWS Security Center.

The process for using Oracle native network encryption with Amazon RDS is as follows:

1. If the DB instance is not associated with an option group that has the network encryption option (NATIVE_NETWORK_ENCRYPTION), you must either modify an existing option group to add the NATIVE_NETWORK_ENCRYPTION option or create a new option group and add the NATIVE_NETWORK_ENCRYPTION option to it. For information about creating or modifying an option group, see Working with Option Groups (p. 510). For information about adding an option to an option group, see Adding an Option to an Option Group (p. 515).
2. Specify the NATIVE_NETWORK_ENCRYPTION option settings for the option group. For information about modifying option settings, see Modifying an Option Setting (p. 519).

   These settings include:

- SQLNET.ENCRYPTION_SERVER–Specifies the encryption behavior when a client, or a server acting as a client, connects to the DB instance. Allowable values are `Accepted`, `Rejected`, `Requested` (the default), and `Required`. `Requested` indicates that the DB instance does not require traffic from the client to be encrypted.
- SQLNET.CRYPTO_CHECKSUM_SERVER–Specifies the data integrity behavior when a client, or a server acting as a client, connects to the DB instance. Allowable values are `Accepted`, `Rejected`, `Requested` (the default), and `Required`. `Requested` indicates that the DB instance does not require the client to perform a checksum.
- SQLNET.ENCRYPTION_TYPES_SERVER–Specifies a list of encryption algorithms used by the DB instance. The DB instance will use each algorithm, in order, to attempt to decrypt the client input until an algorithm succeeds or until the end of the list is reached. Amazon RDS uses the following default list from Oracle. You can change the order or limit the algorithms that the DB instance will accept.
  a. RC4_256: RSA RC4 (256-bit key size)
  b. AES256: AES (256-bit key size)
  c. AES192: AES (192-bit key size)
  d. 3DES168: 3-key Triple-DES (168-bit effective key size)
  e. RC4_128: RSA RC4 (128-bit key size)
  f. AES128: AES (128-bit key size)
  g. 3DES112: 2-key Triple-DES (112-bit effective key size)
  h. RC4_56: RSA RC4 (56-bit key size)
  i. DES: Standard DES (56-bit key size)
  j. RC4_40: RSA RC4 (40-bit key size)
  k. DES40: DES40 (40-bit key size)
- SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER–Specifies the checksum algorithm. The default is sha-1, but md5 is also supported.

3. List the options in the option group to ensure that you have added the NATIVE_NETWORK_ENCRYPTION option and specified the correct settings. You can view the options in an option group using the RDS console, the CLI command rds-describe-option-group-options, or the Amazon RDS API action DescribeOptionGroupOptions.
4. Associate the DB instance with the option group that has the NATIVE_NETWORK_ENCRYPTION option. For information about associating a DB instance with an option group, see Modifying a DB Instance Running the Oracle Database Engine (p. 216).

With Oracle native network encryption, you can also specify network encryption on the client side. On the client (the computer used to connect to the DB instance), you can use the sqlnet.ora file to specify the following client settings: SQLNET.CRYPTO_CHECKSUM_CLIENT , SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT, SQLNET.ENCRYPTION_CLIENT, and SQLNET.ENCRYPTION_TYPES_CLIENT. For information, see Configuring Network Data Encryption and Integrity for Oracle Servers and Clients in the Oracle documentation.

Sometimes, the DB instance will reject a connection request from an application, for example, if there is a mismatch between the encryption algorithms on the client and on the server.

To test Oracle native network encryption , add the following lines to the sqlnet.ora file on the client:

```
DIAG_ADR_ENABLED=off
TRACE_DIRECTORY_CLIENT=/tmp
TRACE_FILE_CLIENT=nettrace
TRACE_LEVEL_CLIENT=16
```

These lines generate a trace file on the client called `/tmp/nettrace*` when the connection is attempted. The trace file contains information on the connection. For more information about connection-related issues when you are using Oracle Native Network Encryption, see About Negotiating Encryption and Integrity in the Oracle documentation.

Oracle Transparent Data Encryption (TDE)

Amazon RDS supports Oracle Transparent Data Encryption (TDE), a feature of the Oracle Advanced Security option available in Oracle Enterprise Edition. This feature automatically encrypts data before it is written to storage and automatically decrypts data when the data is read from storage.

> **Note**
> The **TDE** option is a permanent option that cannot be removed from an option group, and that option group cannot be removed from a DB instance once it is associated with a DB instance. You cannot disable TDE from a DB instance once that instance is associated with an option group with the Oracle TDE option.

Oracle Transparent Data Encryption is used in scenarios where you need to encrypt sensitive data in case data files and backups are obtained by a third party or when you need to address security-related regulatory compliance issues.

A detailed explanation about Oracle Transparent Data Encryption is beyond the scope of this guide. For information about using Oracle Transparent Data Encryption, see Securing Stored Data Using Transparent Data Encryption. For more information about Oracle Advanced Security, see Oracle Advanced Security in the Oracle documentation. For more information on AWS security, see the AWS Security Center.

Oracle Transparent Data Encryption supports two encryption modes: TDE tablespace encryption and TDE column encryption. TDE tablespace encryption is used to encrypt entire application tables. TDE column encryption is used to encrypt individual data elements that contain sensitive data. You can also apply a hybrid encryption solution that uses both TDE tablespace and column encryption. For information about TDE best practices, see Oracle Advanced Security Transparent Data Encryption Best Practices.

Once the option is enabled, you can check the status of the Oracle Wallet by using the following command:

```
SELECT * FROM v$encryption_wallet;
```

To create an encrypted tablespace, use the following command:

```
CREATE TABLESPACE encrypt_ts ENCRYPTION DEFAULT STORAGE (ENCRYPT);
```

To specify the encryption algorithm (for versions 11.2.0.2.v7 or later), use the following command:

```
CREATE TABLESPACE encrypt_ts ENCRYPTION USING 'AES256' DEFAULT STORAGE (ENCRYPT);
```

Note that the previous commands for encrypting a tablespace are the same as the commands you would use with an Oracle installation not on Amazon RDS, and the ALTER TABLE syntax to encrypt a column is also the same as the commands you would use for an Oracle installation not on Amazon RDS.

You should determine if your DB instance is associated with an option group that has the **TDE** option. To view the option group that a DB instance is associated with, you can use the RDS console, the rds-describe-db-instance CLI command, or the API action DescribeDBInstances.

Amazon RDS manages the Oracle Wallet and TDE master key for the DB instance. To comply with several security standards, Amazon RDS is working to implement automatic periodic master key rotation.

The process for using Oracle Transparent Data Encryption (TDE) with Amazon RDS is as follows:

1. If the DB instance is not associated with an option group that has the **TDE** option enabled, you must either create an option group and add the **TDE** option or modify the associated option group to add the **TDE** option. For information about creating or modifying an option group, see Working with Option Groups (p. 510). For information about adding an option to an option group, see Adding an Option to an Option Group (p. 515).
2. Associate the DB instance with the option group with the **TDE** option. For information about associating a DB instance with an option group, see Modifying a DB Instance Running the Oracle Database Engine (p. 216).

If you no longer want to use the TDE option with a DB instance, you must decrypt all your data on the DB instance, copy the data to a new DB instance that is not associated with an option group with TDE enabled, and then delete the original instance. You can rename the new instance to be the same name as the previous DB instance if you prefer.

Using TDE with Data Pump

You can use Oracle Data Pump to import or export encrypted dump files; however. Amazon RDS supports the password encryption mode (ENCRYPTION_MODE=PASSWORD) for Oracle Data Pump. Amazon RDS does not support transparent encryption mode (ENCRYPTION_MODE=TRANSPARENT) for Oracle Data Pump. For more information about using Oracle Data Pump with Amazon RDS, see Oracle Data Pump (p. 219).

Oracle Statspack

The Oracle Statspack option (STATSPACK) installs and enables the Oracle Statspack performance statistics feature. Oracle Statspack is a collection of SQL, PL/SQL, and SQL*Plus scripts that collect, store, and display performance data. For information about using Oracle Statspack, see Oracle Statspack in the Oracle documentation.

> **Note**
> Oracle Statspack is no longer supported by Oracle and has been replaced by the more advanced Automatic Workload Repository (AWR). AWR is available only for Oracle Enterprise Edition customers who have purchased the Diagnostics Pack. Oracle Statspack can be used with any Oracle DB engine on Amazon RDS.

The following steps show you how to work with Oracle Statspack on Amazon RDS:

1. Add the Statspack option to an option group and then associate that option group with your DB instance. Amazon RDS installs the Statspack scripts on the DB instance and then sets up the PERFSTAT user account, the account you use to run the Statspack scripts. If you have installed Statspack, skip this step.

   If you have an existing DB instance that has the PERFSTAT account already created and you want to use Oracle Statspack with it, you must drop the PERFSTAT account before adding the Statspack option to the option group associated with your DB instance. If you attempt to add the Statspack option to an option group associated with a DB instance that already has the PERFSTAT account created, you will get an error and the RDS event RDS-Event-0058 will be generated.

You can drop the PERFSTAT account by running the following command:

```
DROP USER perfstat CASCADE;
```

2. After Amazon RDS has installed Statspack on your DB instance, you must log in to the DB instance using your master user name and master password. You must then reset the PERFSTAT password from the randomly generated value Amazon RDS created when Statspack was installed. After you have reset the PERFSTAT password, you can log in using the PERFSTAT user account and run the Statspack scripts.

   Use the following command to reset the password:

   ```
   ALTER USER perfstat IDENTIFIED BY <new_password> ACCOUNT UNLOCK;
   ```

3. After you have logged on using the PERFSTAT account, you can either manually create a Statspack snapshot or create a job that will take a Statspack snapshot after a given time interval. For example, the following job creates a Statspack snapshot every hour:

   ```
   variable jn number;
   execute dbms_job.submit(:jn, 'statspack.snap;',sysdate,'trunc(SYS
   DATE+1/24,''HH24'')');
   commit;
   ```

4. Once you have created at least two Statspack snapshots, you can view them using the following query:

   ```
   select snap_id, snap_time from stats$snapshot order by 1;
   ```

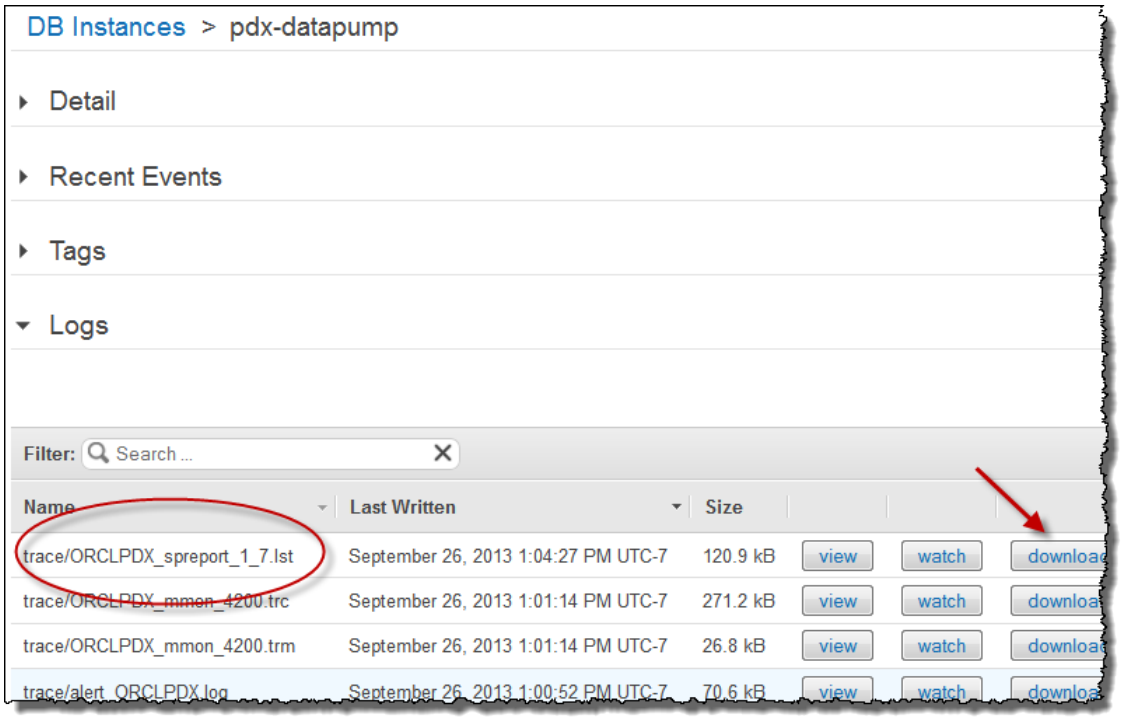
5. To create a Statspack report, you choose two snapshots to analyze and run the following Amazon RDS command:

   ```
   exec RDSADMIN.RDS_RUN_SPREPORT(<begin snap>,<end snap>);
   ```

   For example, the following Amazon RDS command would create a report based on the interval between Statspack snapshots 1 and 7:

   ```
   exec RDSADMIN.RDS_RUN_SPREPORT(1,7);
   ```

The file name of the Statspack report that is generated includes the number of the two Statspack snapshots used. For example, a report file created using Statspack snapshots 1 and 7 would be named ORCL_spreport_1_7.lst. You can download the Statspack report by selecting the report in the Log section of the RDS console and clicking **Download** or you can use the trace file procedures explained in Working with Oracle Trace Files (p. 599).



If an error occurs when producing the report, an error file is created using the same naming conventions but with an extension of .err. For example, if an error occurred while creating a report using Statspack snapshots 1 and 7, the report file would be named ORCL_spreport_1_7.err. You can download the error report by selecting the report in the Log section of the RDS console and clicking **Download** or use the trace file procedures explained in Working with Oracle Trace Files (p. 599).

Oracle Statspack does some basic checking before running the report, so you could also see error messages displayed at the command prompt. For example, if you attempt to generate a report based on an invalid range, such as the beginning Statspack snapshot value is larger than the ending Statspack snapshot value, the error message will be displayed at the command prompt and no error file is created.

```
exec RDSADMIN.RDS_RUN_SPREPORT(2,1);
*
ERROR at line 1:
ORA-20000: Invalid snapshot IDs. Find valid ones in perfstat.stats$snapshot.
```

If you use an invalid number for one of the Statspack snapshots, the error message will also be displayed at the command prompt. For example, if you have 20 Statspack snapshots but request that a report be run using Statspack snapshots 1 and 50, the command prompt will display an error.

```
exec RDSADMIN.RDS_RUN_SPREPORT(1,50);
*
ERROR at line 1:
ORA-20000: Could not find both snapshot IDs
```

For more information about how to use Oracle Statspack, including information on adjusting the amount of data captured by adjusting the snapshot level, go to the Oracle Statspack documentation page.

To remove Oracle Statspack files, use the following command:

```
execute statspack.purge(<begin snap>, <end snap>);
```

Oracle Time Zone

The `Timezone` option lets you change the system time zone used by Oracle databases in a DB instance. You might need to change the time zone for a DB instance if you need to have time compatibility with an on-premises environment or a legacy application. This option changes the time zone at the host level and impacts all date columns and values including `SYSDATE` and `SYSTIMESTAMP`. This option can only be applied once to a DB instance. You should take a DB snapshot of your DB instance before applying this option to a DB instance so that you can recover the instance if the time zone option is set incorrectly.

> **Note**
> Applying the `Timezone` option to option groups used by existing DB instances could cause problems with tables that use system date to add dates or time, so you should analyze your data to determine what impact a time zone change will have. We strongly urge you to test setting this option on a test DB instance before setting it on your production instances.

The `Timezone` option is a permanent and persistent option that cannot be removed from an option group once it is added and the option group cannot be disassociated from a DB instance. This option can be applied immediately by selecting `Apply Immediately` or it can be applied at the next maintenance window.

There are three ways that you can add the `Timezone` option to an option group. You can use the Amazon RDS console, the `rds-add-option-to-option-group` Amazon RDS CLI command, or the `ModifyOptionGroup` API action.

The following example uses the Amazon RDS CLI command `rds-add-option-to-option-group` to add the `Timezone` option and the `TIME_ZONE` option setting to an option group called `myoptiongroup`. The time zone is set to *Asia/Japan*.

```
rds-add-option-to-option-group myoptiongroup --option-name Timezone --settings
 "TIME_ZONE=Asia/Tokyo"
```

The `Timezone` option differs from the `rdsadmin_util.alter_db_time_zone` command. The `rdsadmin_util.alter_db_time_zone` command only changes the time zone for certain data types, while the `Timezone` option changes the time zone at the host level and impacts all date columns and values such as SYSDATE.

The following values can be used for the `TIME_ZONE` option setting:

Africa/Cairo, Africa/Casablanca, Africa/Harare, Africa/Monrovia, Africa/Nairobi, Africa/Tripoli, Africa/Windhoek, America/Araguaina, America/Asuncion, America/Bogota, America/Caracas, America/Chihuahua, America/Cuiaba, America/Denver, America/Fortaleza, America/Guatemala, America/Halifax, America/Manaus, America/Matamoros, America/Monterrey, America/Montevideo, America/Phoenix, America/Santiago, America/Tijuana, Asia/Amman, Asia/Ashgabat, Asia/Baghdad, Asia/Baku, Asia/Bangkok, Asia/Beirut, Asia/Calcutta, Asia/Damascus, Asia/Dhaka, Asia/Irkutsk, Asia/Jerusalem, Asia/Kabul, Asia/Karachi, Asia/Kathmandu, Asia/Krasnoyarsk, Asia/Magadan, Asia/Muscat, Asia/Novosibirsk, Asia/Riyadh, Asia/Seoul, Asia/Shanghai, Asia/Singapore, Asia/Taipei, Asia/Tehran,

Asia/Tokyo, Asia/Ulaanbaatar, Asia/Vladivostok, Asia/Yakutsk, Asia/Yerevan, Atlantic/Azores, Australia/Adelaide, Australia/Brisbane, Australia/Darwin, Australia/Hobart, Australia/Perth, Australia/Sydney, Canada/Newfoundland, Canada/Saskatchewan, Brazil/East, Europe/Amsterdam, Europe/Athens, Europe/Dublin, Europe/Helsinki, Europe/Istanbul, Europe/Kaliningrad, Europe/Moscow, Europe/Paris, Europe/Prague, Europe/Sarajevo, Pacific/Auckland, Pacific/Fiji, Pacific/Guam, Pacific/Honolulu, Pacific/Samoa, US/Alaska, US/Central, US/Eastern, US/East-Indiana, US/Pacific, UTC.

Appendix: Common DBA Tasks for Oracle

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances running the Oracle database engine. In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

For information about working with Oracle log files on Amazon RDS, see Oracle Database Log Files (p. 598)

Tasks

- **System**

  Enabling and disabling Restricted Session (p. 242)

  Flushing the Shared Pool (p. 242)

  Flushing the Buffer Cache (p. 242)

  Disconnecting a Session (for version 11.2.0.3.v1 and later) (p. 243)

  Killing a Session (p. 243)

  Renaming the Global Name (for version 11.2.0.3.v1 and later) (p. 243)

  Granting Privileges to Non-Master Users (p. 244)

  Modifying DBMS_SCHEDULER Jobs (p. 244)
- **Logs**

  Switching Online Log files (p. 245)

  Adding, Dropping and Resizing Online Redo Logs (p. 245)

  Setting Force Logging (for version 11.2.0.3.v1 and later) (p. 248)

  Retaining Archived Redo Logs (for version 11.2.0.2.v7 and later) (p. 248)

  Setting Supplemental Logging (for version 11.2.0.3.v1 and later) (p. 248)

- **Databases**

  Creating and Resizing Tablespaces and Data Files (p. 249)

  Setting Default Tablespace (p. 249)

  Setting Default Temporary Tablespace (p. 249)

  Checkpointing the Database (p. 250)

  Setting Distributed Recovery (for version 11.2.0.3.v1 and later) (p. 250)

  Granting SELECT or EXECUTE privileges to SYS Objects (for version 11.2.0.3.v1 and later) (p. 250)

  Setting the Database Time Zone (p. 251)

  Working with Automatic Workload Repository (AWR) (p. 251)

  Adjusting Database Links for Use with DB Instances in a VPC (p. 251)

Creating New Directories in the Main Data Storage Space (for version 11.2.0.4.v1 and later) (p. 251)

Listing and Reading Files in a DB Instance Directory (for version 11.2.0.3.v1 and later) (p. 252)

Enabling and disabling Restricted Session

Oracle Method	Amazon RDS Method
`alter system enable restricted session;`	`exec rdsadmin.rdsadmin_util.restricted_session(true);`
`alter system disable restricted session;`	`exec rdsadmin.rdsadmin_util.restricted_session(false);`

The following example shows how to enable and disable restricted sessions.

```
select logins from v$instance;

LOGINS
-------
ALLOWED

exec rdsadmin.rdsadmin_util.restricted_session(true);

select logins from v$instance;

LOGINS
----------
RESTRICTED

exec rdsadmin.rdsadmin_util.restricted_session(false);

select logins from v$instance;

LOGINS
-------
ALLOWED
```

Flushing the Shared Pool

Oracle Method	Amazon RDS Method
`alter system flush shared_pool;`	`exec rdsadmin.rdsadmin_util.flush_shared_pool;`

Flushing the Buffer Cache

Oracle Method	Amazon RDS Method
`alter system flush buffer_cache;`	`exec rdsadmin.rdsadmin_util.flush_buffer_cache;`

Disconnecting a Session (for version 11.2.0.3.v1 and later)

The following Amazon RDS method disconnects the current session by ending the dedicated server process. Note that the database must be open to use this method. For more information about disconnecting a session, see the Oracle documentation.

You must specify both the SID and serial number of the session. To obtain these values, query the V$SESSION view. For example, the following query shows all sessions for the user *AWSUSER*:

```
SELECT SID, SERIAL#, STATUS
  FROM V$SESSION
  WHERE USERNAME = 'AWSUSER';
```

Oracle Method	Amazon RDS Method
`alter system disconnect session;`	`exec rdsadmin.rdsadmin_util.disconnect(sid number, serial number, method varchar default 'IMMEDIATE');`

Killing a Session

Oracle Method	Amazon RDS Method

`alter system kill session 'sid, serial#' IMMEDIATE;`	`exec rdsadmin.rdsadmin_util.kill(sid, serial#);`
	
	For use with version 11.2.0.3.v1 or higher:
	`exec rdsadmin.rdsadmin_util.kill(sid number, serial number, method varchar default null);`

If you are using version 11.2.0.3.v1 or higher, you can specify either IMMEDIATE or PROCESS as a value for the `method` parameter. Specifying PROCESS as the `method` value enables you to kill processes associated with a session. You should only do this if killing the session using IMMEDIATE as the `method` value was unsuccessful.

Renaming the Global Name (for version 11.2.0.3.v1 and later)

The following Amazon RDS method changes the global name of the database. Note that the database must be open for the name change to take effect. For more information about changing the global name of a database, see the Oracle documentation.

Oracle Method	Amazon RDS Method
`alter database rename global_name;`	`exec rdsadmin.rdsadmin_util.rename_global_name(p_new_global_name in varchar2);`

Granting Privileges to Non-Master Users

The following example creates a non-master user named *user1* and grants the CREATE SESSION privilege and the SELECT privilege for a database named *sh.sales*:

```
CREATE USER user1 IDENTIFIED BY password;
GRANT CREATE SESSION TO user1;
GRANT SELECT ON sh.sales TO user1;
```

You can grant explicit object privileges for objects in the SYS schema using the SELECT_CATALOG_ROLE and the EXECUTE_CATALOG_ROLE roles. The SELECT_CATALOG_ROLE role allows users SELECT privileges on data dictionary views and the EXECUTE_CATALOG_ROLE role allows users EXECUTE privileges for packages and procedures in the data dictionary.

The following example grants the SELECT_CATALOG_ROLE role to a user named *user1*:

```
GRANT SELECT_CATALOG_ROLE TO user1;
```

The following example grants the EXECUTE_CATALOG_ROLE role to a user named *user1*:

```
GRANT EXECUTE_CATALOG_ROLE TO user1;
```

To view the permissions that the SELECT_CATALOG_ROLE and the EXECUTE_CATALOG_ROLE roles allow, use the following query:

```
SELECT * FROM ROLE_TAB_PRIVS
WHERE ROLE IN ('SELECT_CATALOG_ROLE','EXECUTE_CATALOG_ROLE')
ORDER BY ROLE, TABLE_NAME ASC;
```

Modifying DBMS_SCHEDULER Jobs

You can modify the default DBMS_SCHEDULER jobs and windows by following the Oracle documentation, but you need to prepend the SYS schema name to the WINDOW_NAME. For example, with a local Oracle database you could do the following:

```
execute dbms_scheduler.set_attribute('MONDAY_WINDOW','RESOURCE_PLAN','');
```

For an Amazon RDS DB instance, you would include the SYS schema name:

```
execute dbms_scheduler.set_attribute('SYS.MONDAY_WINDOW','RESOURCE_PLAN','');
```

Switching Online Log files

You can use the following Amazon RDS method to switch log files.

Oracle Method	Amazon RDS Method
`alter system switch logfile;`	`exec rdsadmin.rdsadmin_util.switch_logfile;`

Adding, Dropping and Resizing Online Redo Logs

A newly created Amazon RDS instance using the Oracle database engine will have four 128MB online redo logs. Note that in cases where you want to add more logs, the same restrictions apply to naming physical files as they do for naming online redo logs.

Use the following procedures to add or drop redo logs:

```
exec rdsadmin.rdsadmin_util.add_logfile(size bytes);

exec rdsadmin.rdsadmin_util.drop_logfile(group#);
```

If you are using version 11.2.0.3.v1 or later, you can specify the size modifier as well. For example, the following command would add a 100 Mb log file:

```
exec rdsadmin.rdsadmin_util.add_logfile('100M');
```

Example

The following example shows how you can use the Amazon RDS-provided procedures to resize your online redo logs from their default size to 512M.

```
# Start with four 128m logs.
SQL>select GROUP#, BYTES, STATUS from v$log;

GROUP#      BYTES STATUS
---------- ---------- ----------------
1  134217728 INACTIVE
2  134217728 CURRENT
3  134217728 INACTIVE
4  134217728 INACTIVE

4 rows selected.

# Add four new logs with that are each 512m.

SQL>exec rdsadmin.rdsadmin_util.add_logfile(536870912);

PL/SQL procedure successfully completed.

SQL>exec rdsadmin.rdsadmin_util.add_logfile(536870912);
```

```
PL/SQL procedure successfully completed.

SQL>exec rdsadmin.rdsadmin_util.add_logfile(536870912);

PL/SQL procedure successfully completed.

SQL>exec rdsadmin.rdsadmin_util.add_logfile(536870912);

PL/SQL procedure successfully completed.

# Now query v$log to show that there are 8 logs:

SQL>select GROUP#, BYTES, STATUS from v$log;

GROUP#      BYTES STATUS
---------- ---------- ----------------
1  134217728 INACTIVE
2  134217728 CURRENT
3  134217728 INACTIVE
4  134217728 INACTIVE
5  536870912 UNUSED
6  536870912 UNUSED
7  536870912 UNUSED
8  536870912 UNUSED

8 rows selected.

# Now, drop each INACTIVE log using the group#.

SQL>exec rdsadmin.rdsadmin_util.drop_logfile(1);

PL/SQL procedure successfully completed.

SQL>exec rdsadmin.rdsadmin_util.drop_logfile(3);

PL/SQL procedure successfully completed.

SQL>exec rdsadmin.rdsadmin_util.drop_logfile(4);

PL/SQL procedure successfully completed.

#

SQL>select GROUP#, BYTES, STATUS from v$log;

GROUP#      BYTES STATUS
---------- ---------- ----------------
2  134217728 CURRENT
5  536870912 UNUSED
6  536870912 UNUSED
7  536870912 UNUSED
8  536870912 UNUSED

8 rows selected.

# Switch logs so that group 2 is no longer current:

SQL>exec rdsadmin.rdsadmin_util.switch_logfile;
```

```
PL/SQL procedure successfully completed.

#
SQL>select GROUP#, BYTES, STATUS from v$log;

GROUP#      BYTES STATUS
---------- ---------- ----------------
2  134217728 ACTIVE
5  536870912 CURRENT
6  536870912 UNUSED
7  536870912 UNUSED
8  536870912 UNUSED

5 rows selected.

# Issue a checkpoint to clear log 2

SQL>exec rdsadmin.rdsadmin_util.checkpoint;

PL/SQL procedure successfully completed.

#

SQL>select GROUP#, BYTES, STATUS from v$log;

GROUP#      BYTES STATUS
---------- ---------- ----------------
2  134217728 INACTIVE
5  536870912 CURRENT
6  536870912 UNUSED
7  536870912 UNUSED
8  536870912 UNUSED

5 rows selected.

# Checkpointing clears log group 2 so that its status is now INACTIVE allowing
 us to drop the final log group 2:

SQL>exec rdsadmin.rdsadmin_util.drop_logfile(2);

PL/SQL procedure successfully completed.

# Now, there are four 512m logs. Oracle using Oracle Managed Files (OMF) will
automatically remove the old logfiles from the file system.

SQL>select GROUP#, BYTES, STATUS from v$log;

GROUP#      BYTES STATUS
---------- ---------- ----------------
5  536870912 CURRENT
6  536870912 UNUSED
7  536870912 UNUSED
8  536870912 UNUSED

4 rows selected.
```

Setting Force Logging (for version 11.2.0.3.v1 and later)

The following Amazon RDS method puts the database in or removes the database from FORCE LOGGING mode. In FORCE LOGGING mode, Oracle logs all changes to the database except changes in temporary tablespaces and temporary segments. For more information about forcing logging, see the Oracle documentation.

Oracle Method	Amazon RDS Method
`alter database [no] force logging;`	`exec rdsadmin.rdsadmin_util.force_logging(p_enable in boolean := true);`

Retaining Archived Redo Logs (for version 11.2.0.2.v7 and later)

You can retain archived redo logs on your DB instance for use with products like Oracle LogMiner (DBMS_LOGMNR). Once you have retained the redo logs, you can use LogMiner to analyze the logs as explained in the Oracle documentation. Note that you need to ensure that the DB instance has enough allocated storage to store the retained logs.

Use the Amazon RDS method *rdsadmin.rdsadmin_util.set_configuration* to retain archived redo logs. The following example shows how to retain 24 hours of redo logs:

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
```

If you need to determine how much space your DB instance has used in the last X hours, you can run the following query, replacing X with the number of hours:

```
select sum(blocks * block_size) bytes from v$archived_log where first_time
>=sysdate-X/24 and dest_id=1;
```

Setting Supplemental Logging (for version 11.2.0.3.v1 and later)

The following Amazon RDS method enables supplemental logging, including minimal supplemental logging. Oracle Database does not enable supplemental logging by default. Supplemental logging ensures that LogMiner and products that use LogMiner technology will have sufficient information to support chained rows and various storage arrangements such as cluster tables. For more information on supplemental logging, see the Oracle documentation.

Oracle Method	Amazon RDS Method
`alter database [add|drop] supplemental log;`	`exec rdsadmin.rdsadmin_util.alter_supplemental_logging(p_action in varchar2, p_type in varchar2 default NULL);`
`alter database add supplemental log data (PRIMARY KEY) columns;`	`exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD','PRIMARY KEY');`
`alter database add supplemental log data (ALL) columns;`	`exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD','ALL');`
`alter database add supplemental log data (UNIQUE) columns;`	`exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD','UNIQUE');`

Creating and Resizing Tablespaces and Data Files

Amazon RDS only supports Oracle Managed Files (OMF) for data files, log files and control files. When creating data files and log files you cannot specify physical file names.

The following example creates a tablespace:

```
create tablespace users2;
```

The following example creates temporary tablespace:

```
create temporary tablespace temp01;
```

Because the Oracle `ALTER DATABASE` system privilege is not available on Amazon RDS, you must use `ALTER TABLESPACE` to resize a tablespace. The following example resizes a bigfile tablespace named users2 to 200 MB:

```
alter tablespace users2 resize 200M;
```

For smallfile tablespaces, you need to add an additional datafile, like in the following example:

```
ALTER TABLESPACE users2 ADD DATAFILE SIZE 100000M AUTOEXTEND ON NEXT 250m MAXSIZE
UNLIMITED;
```

Setting Default Tablespace

Oracle Method	Amazon RDS Method
`alter database default tablespace users2;`	`exec rdsadmin.rdsadmin_util.alter_default_tablespace('users2');`

Setting Default Temporary Tablespace

Oracle Method	Amazon RDS Method
`alter database default temporary tablespace temp2;`	`exec rdsadmin.rdsadmin_util.alter_default_temp_tablespace('temp2');`

Checkpointing the Database

Oracle Method	Amazon RDS Method
`alter system checkpoint;`	`exec rdsadmin.rdsadmin_util.checkpoint;`

Setting Distributed Recovery (for version 11.2.0.3.v1 and later)

Oracle Method	Amazon RDS Method
`alter system enable/disable distributed recovery;`	`exec rdsadmin.rdsadmin_util.enable_distr_recovery and rdsadmin_util.disable_distr_recovery (mydatabase);`

Granting SELECT or EXECUTE privileges to SYS Objects (for version 11.2.0.3.v1 and later)

Generally, you can use `grant select_catalog_role` or `grant execute_catalog_role` to grant privileges. If you need to grant privileges to a single object instead of using a role that may contain many objects, you can use the `grant_sys_object` Amazon RDS method. The following procedure transfers existing privileges such as SELECT and EXECUTE via a role to another account. Note that it only grants privileges that the master account already has via a role or direct grant.

Oracle Method	Amazon RDS Method
`grant select on V_$SESSION to myuser;`	`exec rdsadmin.rdsadmin_util.grant_sys_object('V_$SESSION','MYUSER');`

In order to be able to grant privileges on an object, your account must have those privileges granted to it directly with the grant option or via a role granted using WITH ADMIN OPTION. In the most common case, you may want to grant SELECT on a DBA view that has been granted to the SELECT_CATALOG_ROLE role. If that role isn't already directly granted to your user using WITH ADMIN OPTION, then you won't be able to transfer the privilege. If you have the DBA privilege, then you can grant the role directly to another user.

For example, an initial grant for SELECT_CATALOG_ROLE and EXECUTE_CATALOG_ROLE could be:

```
GRANT SELECT_CATALOG_ROLE TO user1 WITH ADMIN OPTION;
GRANT EXECUTE_CATALOG_ROLE TO user1 WITH ADMIN OPTION;
```

In the previous example, since "WITH ADMIN OPTION," was used when granting "user1" access, "user1" will be able to grant access to SYS objects that have been granted to SELECT_CATALOG_ROLE.

Note that objects already granted to PUBLIC do not need to be re-granted, but if you use the grant_sys_object procedure to re-grant access the procedure will not fail. Note too that object names

must be spelled exactly as they appear in DBA_OBJECTS (Most SYS objects are defined in UPPERCASE, so we recommend you try that first).

Setting the Database Time Zone

You can alter the time zone of a database in two ways, by either using the `rdsadmin_util.alter_db_time_zone` command or by setting the Oracle Time Zone (p. 239) option. The `rdsadmin_util.alter_db_time_zone` command changes the time zone for only certain data types and does not change SYSDATE, and must be used with versions 11.2.0.2.v4 or later. The `Timezone` option changes the time zone at the host level and impacts all date columns and values such as SYSDATE.

Oracle Method	Amazon RDS Method
`alter database set time_zone = '+3:00';`	`exec rdsadmin.rdsadmin_util.alter_db_time_zone('+3:00');`

After you alter the time zone, you must reboot the DB instance for the change to take effect.

There are additional restrictions on setting time zones listed in the Oracle documentation.

Working with Automatic Workload Repository (AWR)

If you use Oracle Enterprise Edition and want to use Automatic Workload Repository (AWR), you can enable AWR by changing the `CONTROL_MANAGEMENT_PACK_ACCESS` parameter.

Oracle AWR includes several report generation scripts, such as awrrpt.sql, that are installed on the host server. Since you do not have access to host directories, you can download the scripts from Oracle or by generating them using Oracle Enterprise Manager (OEM).

Adjusting Database Links for Use with DB Instances in a VPC

To use Oracle database links with DB instances inside a VPC, the two instances must be either in the same VPC or you must create an EC2 or VPC security group that both DB instances are a member of. For example, when using Oracle Data Pump and Oracle DBLinks to move data between DB instances, the instances must be members of the same VPC or EC2 security group or they must be in the same VPC. For more information about using database links with Oracle Data Pump, see Oracle Data Pump (p. 219)

Creating New Directories in the Main Data Storage Space (for version 11.2.0.4.v1 and later)

A DB instance come with a set of directories; you can create additional directories using the following Amazon RDS method. The `create_directory()` method lets you create up to 10 directories, all located in your main data storage space. The following example uses the method to create a directory named "MY_DIR".

Oracle Method	Amazon RDS Method
`create directory MY_DIR as '/my/os/pathname';`	`exec rdsadmin.rdsadmin_util.create_directory('MY_DIR');`

You can list the directories by querying the DBA_DIRECTORIES view. Note that the system chose the actual host pathname automatically:

```
select * from DBA_DIRECTORIES where directory_name='MY_DIR';

select directory_path from DBA_DIRECTORIES where directory_name='MY_DIR';

DIRECTORY_PATH
--------------------------------------
/rdsdbdata/userdirs/01
```

The master user name for the DB instance has read and write privileges in the new directory, and can grant access to other users. Note that "execute" privileges are not available for directories on a DB instance. Directories are created in your main data storage space and will consume space and I/O bandwidth.

You can drop a directory that you created by using the Oracle `drop directory` command. Dropping a directory does not remove its contents; because the `create_directory()` method can reuse pathnames, files in dropped directories could appear in a newly created directory. Before you drop a directory, you should use UTL_FILE.FREMOVE to remove files from the directory.

Listing and Reading Files in a DB Instance Directory (for version 11.2.0.3.v1 and later)

You can use the RDSADMIN.RDS_FILE_UTIL.LISTDIR() Amazon RDS method to list the files in any DB instance directory (from DBA_DIRECTORIES) that you have access to:

```
select * from table(RDSADMIN.RDS_FILE_UTIL.LISTDIR('DATA_PUMP_DIR'));
```

If you find a text file that you want to read, you can use the RDSADMIN.RDS_FILE_UTIL.READ_TEXT_FILE() Amazon RDS method. The following example reads the *filename.log* file in the *DATA_PUMP_DIR* directory:

```
select * from table(RDSADMIN.RDS_FILE_UTIL.READ_TEXT_FILE('DATA_PUMP_DIR','filename.log'));
```

Appendix: Using Oracle GoldenGate with Amazon RDS

Oracle GoldenGate is used to collect, replicate, and manage transactional data between databases. It is a log-based change data capture (CDC) and replication software package used with Oracle databases for online transaction processing (OLTP) systems. GoldenGate creates trail files that contain the most recent changed data from the source database and then pushes these files to the target database. You can use Oracle GoldenGate with Amazon RDS for Active-Active database replication, zero-downtime migration and upgrades, disaster recovery, data protection, and in-region and cross-region replication.

Topics

- Setting Up an Oracle GoldenGate Hub on EC2 (p. 256)
- Setting Up a Source Database for Use with GoldenGate on Amazon RDS (p. 258)
- Setting Up a Target Database for Use with GoldenGate on Amazon RDS (p. 262)
- Working with Oracle GoldenGate's Extract and Replicat Utilities (p. 264)
- Troubleshooting Issues When Using Oracle GoldenGate with Amazon RDS (p. 267)

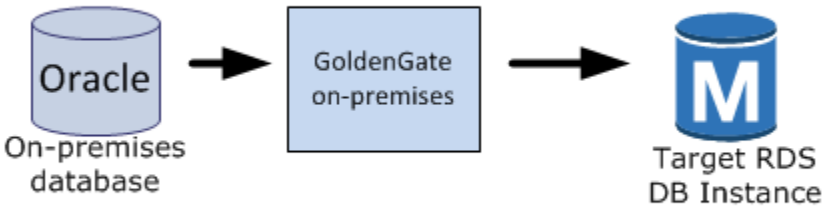
The following are important points to know when working with Oracle GoldenGate on Amazon RDS:

- Oracle GoldenGate with Amazon RDS is available under the “Bring-your-own-license” model in all AWS regions. You are responsible for the set up and management of GoldenGate on Amazon RDS.
- You can use GoldenGate on Amazon RDS with Oracle Database Standard Edition One (SE1), Standard Edition (SE), and Enterprise Edition (EE).
- The Oracle database version must be version 11.2.0.3 or 11.2.0.4, and you must use Oracle GoldenGate version 11.2.1.
- Amazon RDS supports migration and replication across Oracle databases using Oracle GoldenGate. We do not support nor prevent customers from migrating or replicating across heterogeneous databases.
- You can use GoldenGate on Amazon RDS Oracle DB instances that use Oracle Transparent Data Encryption (TDE). Since trail files save data unencrypted by default, you should encrypt the pipeline between the source instance, the GoldenGate hub, and the target instance using `sqlnet.ora` encryption. For more information on `sqlnet.ora` encryption, see the Oracle documentation.
- Oracle GoldenGate DDL is not currently supported.

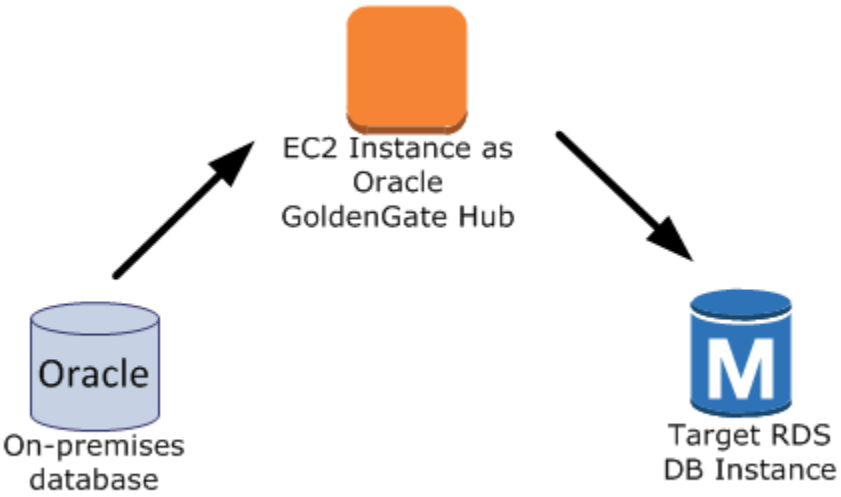
The Oracle GoldenGate architecture for use with Amazon RDS consists of three decoupled modules. The source database can be either an on-premises Oracle database, an Oracle database on an EC2 instance, or an Oracle database on an Amazon RDS DB instance. Next, the GoldenGate hub, which moves transaction information from the source database to the target database, can be either an EC2 instance with Oracle Database 11.2.0.3 or 11.2.0.4 and with GoldenGate 11.2.1 installed, or an on-premises Oracle installation. You can have more than one EC2 hub, and we recommend that you use two hubs if you are using GoldenGate for cross-region replication. Finally, the target database can be either on an Amazon RDS DB instance, on an EC2 instance, or on an on-premises location.

Oracle GoldenGate on Amazon RDS supports the following common scenarios:

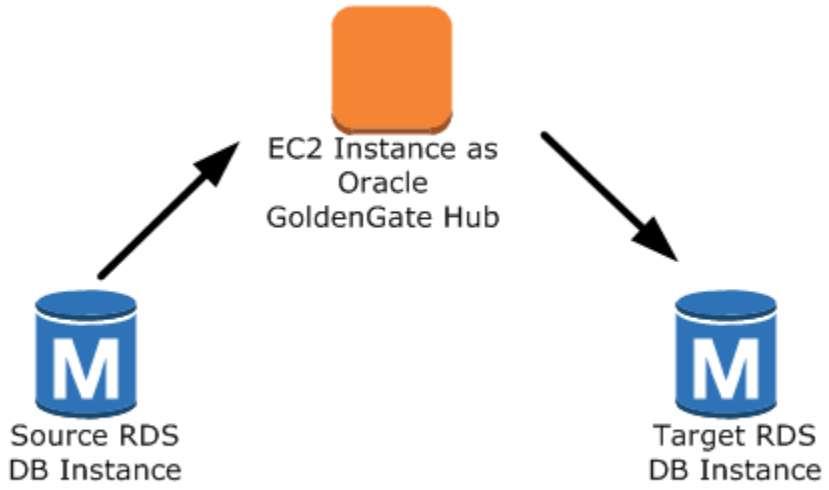
Scenario 1: An on-premises Oracle source database and on-premises Oracle GoldenGate hub, that provides data to a target Amazon RDS DB instance



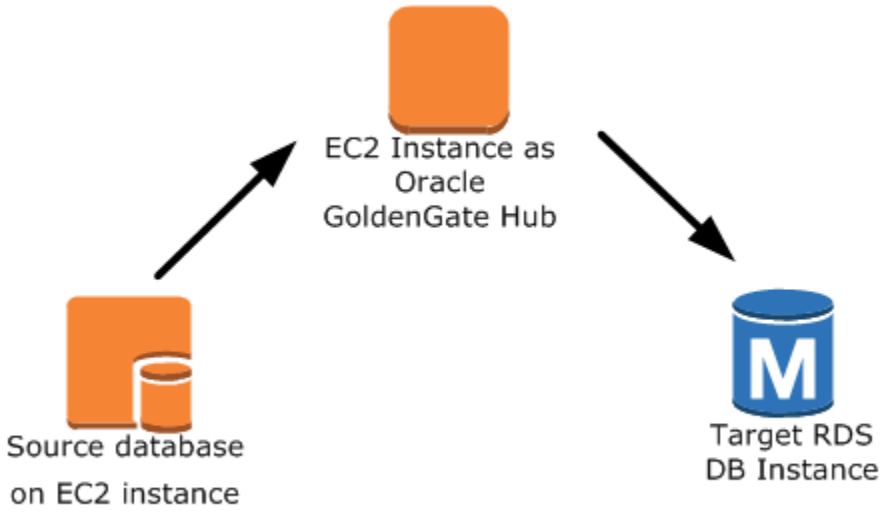
Scenario 2: An on-premises Oracle database that acts as the source database, connected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance



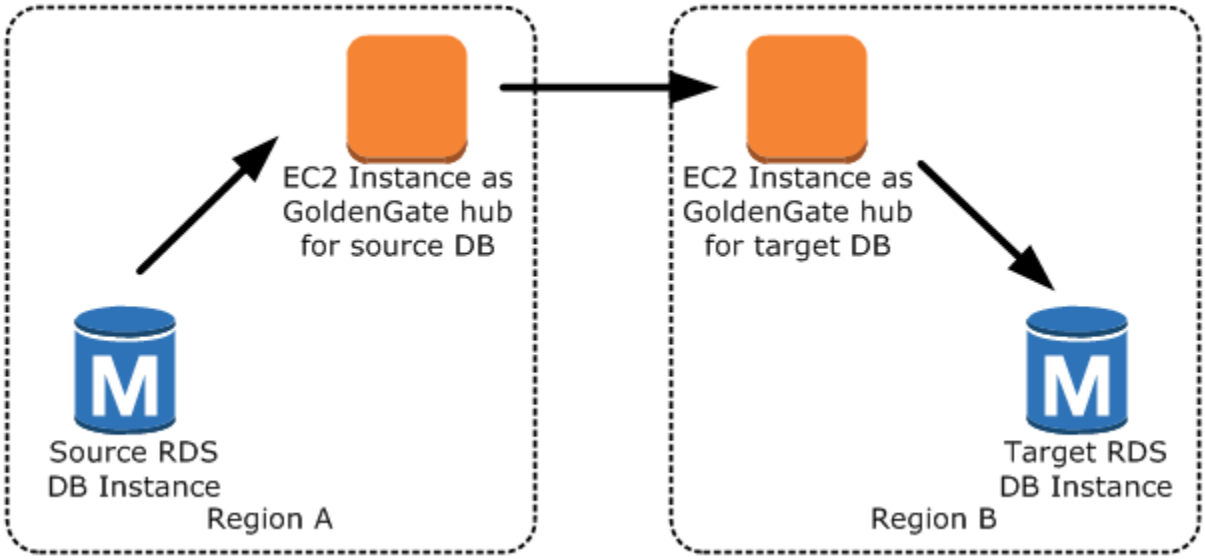
Scenario 3: An Oracle database on an Amazon RDS DB instance that acts as the source database, connected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance



Scenario 4: An Oracle database on an Amazon EC2 instance that acts as the source database, connected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance



Scenario 5: An Oracle database on an Amazon RDS DB instance connected to an Amazon EC2 instance hub in the same region, connected to an Amazon EC2 instance hub in a different region that provides data to the target Amazon RDS DB instance in the same region as the second EC2 instance hub.



Note
Any issues that impact running Oracle GoldenGate on an on-premises environment will also impact running GoldenGate on AWS. We strongly recommend that you monitor the GoldenGate hub to ensure that *Extract* and *Replicat* are resumed if a failover occurs. Since the GoldenGate hub is run on an Amazon EC2 instance, Amazon RDS does not manage the GoldenGate hub and cannot ensure that it is running.

You can use GoldenGate using Amazon RDS to upgrade to major versions of Oracle. For example, you can use GoldenGate using Amazon RDS to upgrade from an Oracle version 8 on-premises database to an Oracle database running version 11.2.0.3 or 11.2.0.4 on an Amazon RDS DB instance.

To set up Oracle GoldenGate using Amazon RDS, you configure the hub on the EC2 instance, and then configure the source and target databases. The following steps show how to set up GoldenGate for use with Amazon RDS. Each step is explained in detail in the following sections:

- Setting Up an Oracle GoldenGate Hub on EC2 (p. 256)
- Setting Up a Source Database for Use with GoldenGate on Amazon RDS (p. 258)
- Setting Up a Target Database for Use with GoldenGate on Amazon RDS (p. 262)
- Working with Oracle GoldenGate's Extract and Replicat Utilities (p. 264)

Setting Up an Oracle GoldenGate Hub on EC2

There are several steps to creating an Oracle GoldenGate hub on an Amazon EC2 instance. First, you create an EC2 instance with a full installation of Oracle DBMS 11g version 11.2.0.3 or 11.2.0.4. The EC2 instance must also have Oracle GoldenGate 11.2.1 software installed, and you must have Oracle patch 13328193 installed. For more information about installing GoldenGate, see the Oracle documentation.

Since the EC2 instance that is serving as the GoldenGate hub stores and processes the transaction information from the source database into trail files, you must have enough allocated storage to store the trail files. You must also ensure that the EC2 instance has enough processing power to manage the amount of data being processed and enough memory to store the transaction information before it is written to the trail file.

The following tasks set up a GoldenGate hub on an Amazon EC2 instance; each task is explained in detail in this section. The tasks include:

- Add an alias to the tnsname.ora file
- Create the GoldenGate subdirectories
- Update the GLOBALS parameter file
- Configure the mgr.prm file and start the *manager*

Add the following entry to the tnsname.ora file to create an alias. For more information on the tnsname.ora file, see the Oracle documentation.

```
$ cat /example/config/tnsnames.ora
TEST=
(DESCRIPTION=
 (ENABLE=BROKEN)
 (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=TCP)(HOST=goldengate-test.abcdef12345.us-west-
2.rds.amazonaws.com)(PORT=8200))
 )
 (CONNECT_DATA=
   (SID=ORCL)
 )
)
```

Next, create subdirectories in the GoldenGate directory using the EC2 command line shell and *ggsci*, the GoldenGate command interpreter. The subdirectories are created under the gg directory and include directories for parameter, report, and checkpoint files.

```
prompt$ cd /gg
prompt$ ./ggsci
  GGSCI> CREATE SUBDIRS
```

Create a GLOBALS parameter file using the EC2 command line shell. Parameters that affect all GoldenGate processes are defined in the GLOBALS parameter file. The following example creates the necessary file:

```
prompt$ cd $GGHOME
prompt$ vi GLOBALS
CheckpointTable oggadm1.oggchkpt
```

The last step in setting up and configuring the GoldenGate hub is to configure the *manager*. Add the following lines to the *mgr.prm* file, then start the *manager* using *ggsci*:

```
PORT 8199
PurgeOldExtracts ./dirdat/*, UseCheckpoints, MINKEEPDAYS 5
```

```
GGSCI>  start mgr
```

Once you have completed these steps, the GoldenGate hub is ready for use. Next, you set up the source and target databases.

Setting Up a Source Database for Use with GoldenGate on Amazon RDS

There are several differences in the set up steps between a source database running Oracle version 11.2.0.3 and version 11.2.0.4. See the appropriate version for the correct set up steps.

Topics

- For Source Databases Running Oracle 11.2.0.3 (p. 258)
- For Source Databases Running Oracle 11.2.0.4 or Later (p. 260)

For Source Databases Running Oracle 11.2.0.3

The following tasks set up a source database running version 11.2.0.3 for use with GoldenGate; each task is explained in detail in this section. The tasks include:

- Set the `compatible` parameter to 11.2.0.3.
- Enable supplemental logging.
- Set the retention period for archived redo logs for the GoldenGate source database.
- Create a GoldenGate user account on the source database.
- Grant the necessary privileges to the GoldenGate user.

The source database must have the `compatible` parameter set to 11.2.0.3. If you are using an Oracle database on an Amazon RDS DB instance as the source database, you must have a parameter group with the `compatible` parameter set to 11.2.0.3 associated with the DB instance. If you change the `compatible` parameter in a parameter group associated with the DB instance, the change requires an instance reboot. You can use the following Amazon RDS CLI commands to create a new parameter group and set the `compatible` parameter. Note that you must associate the new parameter group with the source DB instance:

```
rds-create-db-parameter-group example-goldengate -d "Parameters to allow
GoldenGate" -f oracle-ee-11.2
rds-modify-db-parameter-group example-goldengate -p "name=compatible,
value=11.2.0.3, method=pending-reboot"
rds-modify-db-instance example-test -g example-goldengate [dash dash]apply-im
mediately
rds-reboot-db-instance example-test
```

Always retain the parameter group with the `compatible` parameter. If you restore an instance from a DB snapshot, you must modify the restored instance to use the parameter group that has a matching or greater `compatible` parameter value. This should be done as soon as possible after the restore action and will require a reboot of the instance.

The source database must have the supplemental logging parameter enabled. If you are using an Oracle database on an Amazon RDS DB instance as the source database, you can use the following Amazon RDS procedures to enable supplemental logging:

```
exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD');
exec rdsadmin.rdsadmin_util.force_logging(true);
exec rdsadmin.rdsadmin_util.switch_logfile;
```

The source database must also retain archived redo logs. For example, the following command sets the retention period for archived redo logs to 24 hours:

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
```

The duration for log retention is specified in hours. The duration should exceed any potential downtime of the source instance or any potential communication/networking issues to the source instance, so that Oracle GoldenGate can recover logs from the source instance as needed. The absolute minimum value required is one (1) hour of logs retained.

A log retention setting that is too small will result in the following message:

```
ERROR OGG-02028  Failed to attach to logmining server OGG$<extract_name> error
 26927 - ORA-26927: altering an outbound server with a remote capture is not
allowed.
```

Because these logs are retained on your DB instance, you need to ensure that you have enough storage available on your instance to accommodate the log files. To see how much space you have used in the last "X" hours, use the following query, replacing "X" with the number of hours.

```
select sum(blocks * block_size) bytes from v$archived_log where
next_time>=sysdate-X/24 and dest_id=1;
```

GoldenGate runs as a database user and must have the appropriate database privileges to access the redo and archive logs for the source database, so you must create a GoldenGate user account on the source database. For more information about the permissions for a GoldenGate user account, see the sections 4, section 4.4, and table 4.1 in the Oracle documentation.

The following statements create a user account named *oggadm1*:

```
CREATE tablespace administrator;
CREATE USER oggadm1  IDENTIFIED BY "XXXXXX"  default tablespace ADMINISTRATOR
temporary tablespace TEMP;
```

Finally, grant the necessary privileges to the GoldenGate user account. The following statements grant privileges to a user named *oggadm1*:

```
grant create session, alter session to oggadm1;
grant resource to oggadm1;
grant select any dictionary to oggadm1;
grant flashback any table to oggadm1;
grant select any table to oggadm1;
grant select_catalog_role to <RDS instance master username> with admin option;
exec RDSADMIN.RDSADMIN_UTIL.GRANT_SYS_OBJECT ('DBA_CLUSTERS', 'OGGADM1');
grant execute on dbms_flashback to oggadm1;
grant select on SYS.v_$database to oggadm1;
grant alter any table to oggadm1;

EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE (grantee=>'OGGADM1',priv
ilege_type=>'capture',grant_select_privileges=>true, do_grants=>TRUE);
```

For Source Databases Running Oracle 11.2.0.4 or Later

When your source database is running version 11.2.0.4 or later, there are three tasks you need to accomplish to set up a source database for use with GoldenGate:

- Set the `compatible` parameter to 11.2.0.4 or later.
- Set the `ENABLE_GOLDENGATE_REPLICATION` parameter to *True*. This parameter turns on supplemental logging for the source database. If your source database is on an Amazon RDS DB instance, you must have a parameter group assigned to the DB instance with the `ENABLE_GOLDENGATE_REPLICATION` parameter set to *true*. For more information about the `ENABLE_GOLDENGATE_REPLICATION` parameter, see the Oracle documentation.
- Set the retention period for archived redo logs for the GoldenGate source database.
- Create a GoldenGate user account on the source database.
- Grant the necessary privileges to the GoldenGate user.

The source database must have the `compatible` parameter set to 11.2.0.4 or later. If you are using an Oracle database on an Amazon RDS DB instance as the source database, you must have a parameter group with the `compatible` parameter set to 11.2.0.4 or later associated with the DB instance. If you change the `compatible` parameter in a parameter group associated with the DB instance, the change requires an instance reboot. You can use the following Amazon RDS CLI commands to create a new parameter group and set the `compatible` parameter. Note that you must associate the new parameter group with the source DB instance:

```
rds-create-db-parameter-group example-goldengate -d "Parameters to allow
GoldenGate" -f oracle-ee-11.2
rds-modify-db-parameter-group example-goldengate -p "name=compatible,
value=11.2.0.4, method=pending-reboot"
rds-modify-db-instance example-test -g example-goldengate --apply-immediately
rds-reboot-db-instance example-test
```

Always retain the parameter group with the `compatible` parameter. If you restore an instance from a DB snapshot, you must modify the restored instance to use the parameter group that has a matching or greater `compatible` parameter value. This should be done as soon as possible after the restore action and will require a reboot of the instance.

The `ENABLE_GOLDENGATE_REPLICATION` parameter, when set to *True*, turns on supplemental logging for the source database and configures the required GoldenGate permissions. If your source database is on an Amazon RDS DB instance, you must have a parameter group assigned to the DB instance with the `ENABLE_GOLDENGATE_REPLICATION` parameter set to *true*. For more information about the `ENABLE_GOLDENGATE_REPLICATION` parameter, see the Oracle documentation.

The source database must also retain archived redo logs. For example, the following command sets the retention period for archived redo logs to 24 hours:

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
```

The duration for log retention is specified in hours. The duration should exceed any potential downtime of the source instance or any potential communication/networking issues to the source instance, so that Oracle GoldenGate can recover logs from the source instance as needed. The absolute minimum value required is one (1) hour of logs retained.

A log retention setting that is too small will result in the following message:

```
ERROR OGG-02028  Failed to attach to logmining server OGG$<extract_name> error
 26927 - ORA-26927: altering an outbound server with a remote capture is not
allowed.
```

Because these logs are retained on your DB instance, you need to ensure that you have enough storage available on your instance to accommodate the log files. To see how much space you have used in the last "X" hours, use the following query, replacing "X" with the number of hours.

```
select sum(blocks * block_size) bytes from v$archived_log where
next_time>=sysdate-X/24 and dest_id=1;
```

GoldenGate runs as a database user and must have the appropriate database privileges to access the redo and archive logs for the source database, so you must create a GoldenGate user account on the source database. For more information about the permissions for a GoldenGate user account, see the sections 4, section 4.4, and table 4.1 in the Oracle documentation.

The following statements create a user account named *oggadm1*:

```
CREATE tablespace administrator;
CREATE USER oggadm1  IDENTIFIED BY "XXXXXX"  default tablespace ADMINISTRATOR
temporary tablespace TEMP;
```

Finally, grant the necessary privileges to the GoldenGate user account. The following statements grant privileges to a user named *oggadm1*:

```
grant create session, alter session to oggadm1;
grant resource to oggadm1;
grant select any dictionary to oggadm1;
grant flashback any table to oggadm1;
grant select any table to oggadm1;
grant select_catalog_role to <RDS instance master username> with admin option;
exec RDSADMIN.RDSADMIN_UTIL.GRANT_SYS_OBJECT ('DBA_CLUSTERS', 'OGGADM1');
grant execute on dbms_flashback to oggadm1;
grant select on SYS.v_$database to oggadm1;
grant alter any table to oggadm1;

EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE (grantee=>'OGGADM1',priv
ilege_type=>'capture',grant_select_privileges=>true, do_grants=>TRUE);
```

Setting Up a Target Database for Use with GoldenGate on Amazon RDS

Oracle has recently simplified the set up for using GoldenGate. For example, the configuration tasks to set up GoldenGate support for version 12.1.0.1 is the same as the configuration tasks to set up GoldenGate support for version 11.2.0.3. Similarly, the configuration tasks to set up GoldenGate support for version 12.1.0.2 is the same as the configuration tasks to set up GoldenGate support for version 11.2.0.4.

Target Databases Setup for Running GoldenGate on Oracle 11.2.0.3 or Oracle 12.1.0.1

The following tasks set up a target DB instance for use with GoldenGate:

- Set the `compatible` parameter to 11.2.0.3 or later
- Create and manage a GoldenGate user account on the target database
- Grant the necessary privileges to the GoldenGate user

GoldenGate runs as a database user and must have the appropriate database privileges, so you must create a GoldenGate user account on the target database. The following statements create a user named *oggadm1*:

```
create tablespace administrator;
create tablespace administrator_idx;
CREATE USER oggadm1  IDENTIFIED BY "XXXXXX"  default tablespace ADMINISTRATOR
temporary tablespace TEMP;
alter user oggadm1 quota unlimited on ADMINISTRATOR;
alter user oggadm1 quota unlimited on ADMINISTRATOR_IDX;
```

Finally, grant the necessary privileges to the GoldenGate user account. The following statements grant privileges to a user named *oggadm1*:

```
grant create session        to oggadm1;
grant alter session         to oggadm1;
grant CREATE CLUSTER        to oggadm1;
grant CREATE INDEXTYPE      to oggadm1;
grant CREATE OPERATOR       to oggadm1;
grant CREATE PROCEDURE      to oggadm1;
grant CREATE SEQUENCE       to oggadm1;
grant CREATE TABLE          to oggadm1;
grant CREATE TRIGGER        to oggadm1;
grant CREATE TYPE           to oggadm1;
grant select any dictionary to oggadm1;
grant create any table      to oggadm1;
grant alter any table       to oggadm1;
grant lock any table        to oggadm1;
grant select any table      to oggadm1;
grant insert any table      to oggadm1;
grant update any table      to oggadm1;
grant delete any table      to oggadm1;

EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE
(grantee=>'OGGADM1',privilege_type=>'apply',
grant_select_privileges=>true, do_grants=>TRUE);
```

Target Databases Setup for Running GoldenGate on Oracle 11.2.0.4, Oracle 12.1.0.2 or Later

The following tasks set up a target DB instance for use with GoldenGate:

- Set the `compatible` parameter to 11.2.0.4 or later
- Set the ENABLE_GOLDENGATE_REPLICATION parameter to *True*. If your target database is on an Amazon RDS DB instance, you must have a parameter group assigned to the DB instance with the ENABLE_GOLDENGATE_REPLICATION parameter set to *true*. For more information about the ENABLE_GOLDENGATE_REPLICATION parameter, see the Oracle documentation .
- Create and manage a GoldenGate user account on the target database
- Grant the necessary privileges to the GoldenGate user

GoldenGate runs as a database user and must have the appropriate database privileges, so you must create a GoldenGate user account on the target database. The following statements create a user named *oggadm1*:

```
create tablespace administrator;
create tablespace administrator_idx;
CREATE USER oggadm1  IDENTIFIED BY "XXXXXX" default
tablespace ADMINISTRATOR temporary tablespace TEMP;
alter user oggadm1 quota unlimited on ADMINISTRATOR;
alter user oggadm1 quota unlimited on ADMINISTRATOR_IDX;
```

Finally, grant the necessary privileges to the GoldenGate user account. The following statements grant privileges to a user named *oggadm1*:

```
grant create session        to oggadm1;
grant alter session         to oggadm1;
grant CREATE CLUSTER        to oggadm1;
grant CREATE INDEXTYPE      to oggadm1;
grant CREATE OPERATOR       to oggadm1;
grant CREATE PROCEDURE      to oggadm1;
grant CREATE SEQUENCE       to oggadm1;
grant CREATE TABLE          to oggadm1;
grant CREATE TRIGGER        to oggadm1;
grant CREATE TYPE           to oggadm1;
grant select any dictionary to oggadm1;
grant create any table      to oggadm1;
grant alter any table       to oggadm1;
grant lock any table        to oggadm1;
grant select any table      to oggadm1;
grant insert any table      to oggadm1;
grant update any table      to oggadm1;
grant delete any table      to oggadm1;

EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE
(grantee=>'OGGADM1',privilege_type=>'apply',
grant_select_privileges=>true, do_grants=>TRUE);
```

Working with Oracle GoldenGate's Extract and Replicat Utilities

The Oracle GoldenGate utilities *Extract* and *Replicat* work together to keep the source and target databases in sync via incremental transaction replication using trail files. All changes that occur on the source database are automatically detected by *Extract*, then formatted and transferred to trail files on the GoldenGate on-premises or EC2-instance hub. After initial load is completed, the data is read from these files and replicated to the target database by the *Replicat* utility.

Running Oracle GoldenGate's Extract Utility

The *Extract* utility retrieves, converts, and outputs data from the source database to trail files. *Extract* queues transaction details to memory or to temporary disk storage. When the transaction is committed to the source database, *Extract* flushes all of the transaction details to a trail file for routing to the GoldenGate on-premises or EC2-instance hub and then to the target database.

The following tasks enable and start the *Extract* utility:

- Configure the *Extract* parameter file on the GoldenGate hub (on-premises or EC2 instance). The following listing shows an example *Extract* parameter file.

```
EXTRACT EABC
SETENV (ORACLE_SID=ORCL)
SETENV (NLSLANG=AL32UTF8)

USERID oggadm1@TEST, PASSWORD XXXXXX
EXTTRAIL /path/to/goldengate/dirdat/ab

IGNOREREPLICATES
```

```
GETAPPLOPS
TRANLOGOPTIONS EXCLUDEUSER OGGADM1

TABLE EXAMPLE.TABLE;
```

- On the GoldenGate hub, launch the GoldenGate command line interface (*ggsci*). Log into the source database. The following example shows the format for logging in:

```
dblogin userid <user>@<db tnsname>
```

- Add a checkpoint table for the database:

```
add checkpointtable
```

- Add transdata to turn on supplemental logging for the database table:

```
add trandata <user>.<table>
```

  Alternatively, you can add transdata to turn on supplemental logging for all tables in the database:

```
add trandata <user>.*
```

- Using the *ggsci* command line, enable the *Extract* utility using the following commands:

```
add extract <extract name> tranlog, INTEGRATED tranlog, begin now
add exttrail <path-to-trail-from-the param-file> extract <extractname-from-
paramfile>, MEGABYTES Xm
```

- Register the *Extract* utility with the database so that the archive logs are not deleted. This allows you to recover old, uncommitted transactions if necessary. To register the *Extract* utility with the database, use the following command:

```
register EXTRACT <extract process name>, DATABASE
```

- To start the *Extract* utility, use the following command:

```
start <extract process name>
```

Running Oracle GoldenGate's Replicat Utility

The *Replicat* utility is used to "push" transaction information in the trail files to the target database.

The following tasks enable and start the *Replicat* utility:

- Configure the *Replicat* parameter file on the GoldenGate hub (on-premises or EC2 instance). The following listing shows an example *Replicat* parameter file.

```
REPLICAT RABC
SETENV (ORACLE_SID=ORCL)
SETENV (NLSLANG=AL32UTF8)

USERID oggadm1@TARGET, password XXXXXX

ASSUMETARGETDEFS
MAP EXAMPLE.TABLE, TARGET EXAMPLE.TABLE;
```

- Launch the GoldenGate command line interface (ggsci). Log into the target database. The following example shows the format for logging in:

```
dblogin userid <user>@<db tnsname>
```

- Using the *ggsci* command line, add a checkpoint table. Note that the user indicated should be the GoldenGate user account, not the target table schema owner. The following example creates a checkpoint table named *gg_checkpoint*.

```
add checkpointtable <user>.gg_checkpoint
```

- To enable the *replicat* utility, use the following command:

```
add replicat <replicat name> EXTTRAIL <extract trail file> CHECKPOINTTABLE
<user>.gg_checkpoint
```

- To start the *replicat* utility, use the following command:

```
start <replicat name>
```

Troubleshooting Issues When Using Oracle GoldenGate with Amazon RDS

This section explains the most common issues when using GoldenGate with Amazon RDS.

Topics

- Using GoldenGate with Amazon EC2 Instances (p. 267)
- Log Retention (p. 267)
- GoldenGate appears to be properly configured but replication is not working (p. 268)

Using GoldenGate with Amazon EC2 Instances

If you are using GoldenGate with an EC2 instance, the EC2 instance must have a full installation of Oracle DBMS 11g version 11.2.0.4. The EC2 instance must also have Oracle GoldenGate 11.2.1 installed, and you must have Oracle patch 13328193 installed. If you do not have these items correctly installed, you will see this error message:

```
2014-03-06 07:09:21  ERROR   OGG-02021  This database lacks the required librar
ies to support integrated capture.
```

To determine what patches you currently have installed, run the command **opatch lsinventory** on your EC2 instance.

Log Retention

You must have log retention enabled. If you do not, or if the retention value is too small, you will see the following message:

```
2014-03-06 06:17:27  ERROR   OGG-00446  error 2 (No such file or directory)
```

```
opening redo log /rdsdbdata/db/GGTEST3_A/onlinelog/o1_mf_2_9k4bp1n6_.log
for sequence 1306Not able to establish initial position for begin time 2014-03-
06 06:16:55.
```

GoldenGate appears to be properly configured but replication is not working

For pre-existing tables, GoldenGate needs to be told which SCN it should work from. Take the following steps to fix this issue:

- Launch the GoldenGate command line interface (ggsci). Log into the source database. The following example shows the format for logging in:

```
dblogin userid <user>@<db tnsname>
```

- Using the ggsci command line, set up the start SCN for the extract process. The following example sets the SCN to 223274 for the extract:

```
ALTER EXTRACT <extract process name> SCN 223274
start <extract process name>
```

- Log into the target database. The following example shows the format for logging in:

```
dblogin userid <user>@<db tnsname>
```

- Using the ggsci command line, set up the start SCN for the replicat process. The following example sets the SCN to 223274 for the replicat:

```
start <replicat process name> atcsn 223274
```

Appendix: Using AWS CloudHSM to Store Amazon RDS Oracle TDE Keys

AWS CloudHSM is a service that lets you use a hardware appliance called a hardware security module (HSM) that provides secure key storage and cryptographic operations. You can use AWS CloudHSM with an Oracle Enterprise Edition DB instance to store TDE keys when using Oracle Transparent Data Encryption (TDE). You enable an Amazon RDS DB instance to use AWS CloudHSM by setting up an HSM appliance, setting the proper permissions for cross-service access, and then setting up Amazon RDS and the DB instance that will use AWS CloudHSM.

The number of Oracle databases you can support on a single CloudHSM partition will depend on the rotation schedule you choose for your data. You should rotate your keys as often as your data needs require. The PCI-DSS documentation and the National Institute of Standards and Technology (NIST) provide guidance on appropriate key rotation frequency. You can maintain approximately 10,000 symmetric master keys per CloudHSM device. Note that after key rotation the old master key remains on the partition and is still counted against the per-partition maximum.

AWS CloudHSM works with Amazon Virtual Private Cloud (Amazon VPC). An appliance is provisioned inside your VPC with a private IP address that you specify, providing simple and private network connectivity to your Amazon RDS DB instance. Your HSM appliances are dedicated exclusively to you and are isolated from other AWS customers. For more information about working with Amazon VPC and Amazon RDS, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73) and Creating a DB Instance in a VPC (p. 566).

Important
This document tells you how to install and use AWS CloudHSM with an Amazon RDS Oracle DB instance that is using Oracle TDE encryption. Review the following availability and pricing information before you setup AWS CloudHSM:

- US East (N. Virginia), US West (Oregon), EU (Ireland), Asia Pacific (Sydney), EU (Frankfurt), Asia Pacific (Singapore), and Asia Pacific (Tokyo) regions.
- AWS CloudHSM pricing and free trial:

  CloudHSM pricing information is available on the CloudHSM pricing page. If you want to try the CloudHSM service for free, please review the free trial page for more information.
- CloudHSM upfront fee refund (CLI Tools):

  Please note that there is an upfront fee charged for each new CloudHSM instance you create using the `create-hsm` CLI command. If you accidentally provision a CloudHSM device and want to request a refund, please delete the CloudHSM instance using the `delete-hsm` command, and then go to the AWS Support Center, create a new case, and then select **Account and Billing Support**.
- CloudHSM upfront fee refund (API):

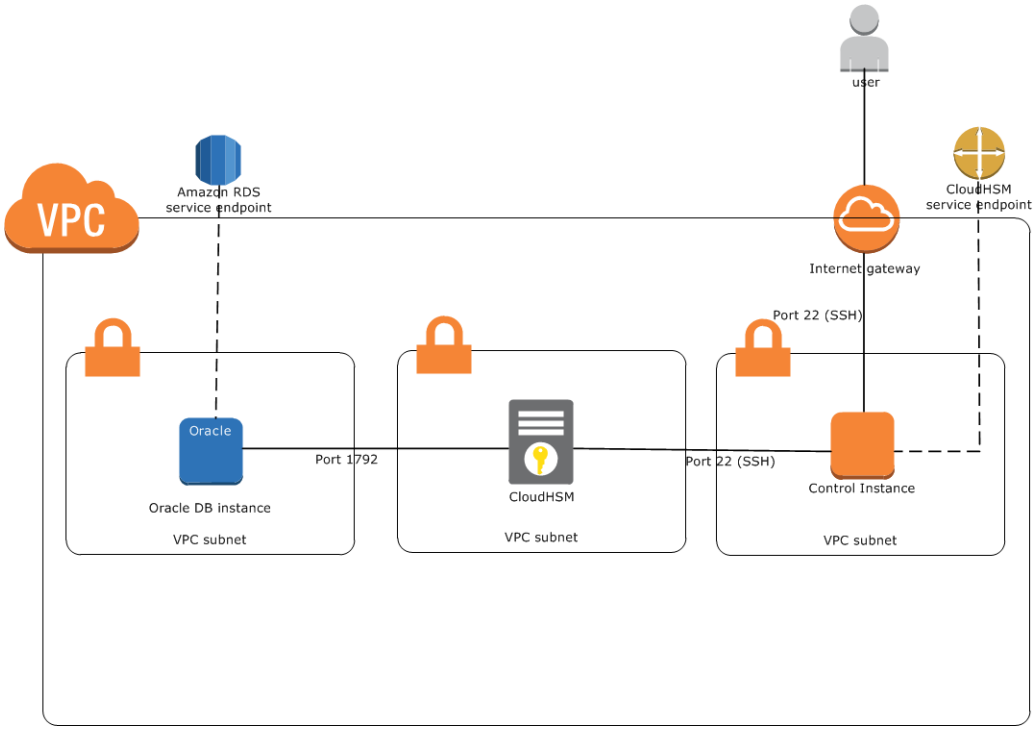
  Please note that there is an upfront fee charged for each new CloudHSM instance you create using the `CreatHSM` API method. If you accidentally provision a CloudHSM device and want to request a refund, please delete the CloudHSM instance using the `DeleteHSM` API method, and then go to the AWS Support Center, create a new case, and then select **Account and Billing Support**.

To use AWS CloudHSM with an Amazon RDS Oracle DB instance, you must complete the following tasks, which are explained in detail in the following sections:

- Setting Up AWS CloudHSM to Work with Amazon RDS (p. 270)
- Setting Up Amazon RDS to Work with AWS CloudHSM (p. 274)

When you complete the entire setup, you should have the following AWS components.

- An AWS CloudHSM control instance that will communicate with the HSM appliance using port 22, and the AWS CloudHSM endpoint. The AWS CloudHSM control instance is an EC2 instance that is in the same VPC as the HSMs and is used to manage the HSMs.
- An Amazon RDS Oracle DB instance that will communicate with the Amazon RDS service endpoint, as well as the HSM appliance using port 1792.



Topics

- Setting Up AWS CloudHSM to Work with Amazon RDS (p. 270)
- Setting Up Amazon RDS to Work with AWS CloudHSM (p. 274)
- Verifying the HSM Connection, the Oracle Keys in the HSM, and the TDE Key (p. 280)
- Restoring Encrypted DB Instances (p. 282)
- Managing a Multi-AZ Failover (p. 283)

Setting Up AWS CloudHSM to Work with Amazon RDS

To use AWS CloudHSM with an Oracle DB instance using TDE, you must first complete the tasks required to setup AWS CloudHSM. The tasks are explained in detail in the following sections. These tasks include:

Topics

- Completing the AWS CloudHSM Prerequisites (p. 271)
- Installing the AWS CloudHSM Command Line Interface Tools (p. 271)
- Configuring Your HSMs (p. 271)
- Creating Your High-Availability Partition Group (p. 272)

- Password Worksheet (p. 273)

Completing the AWS CloudHSM Prerequisites

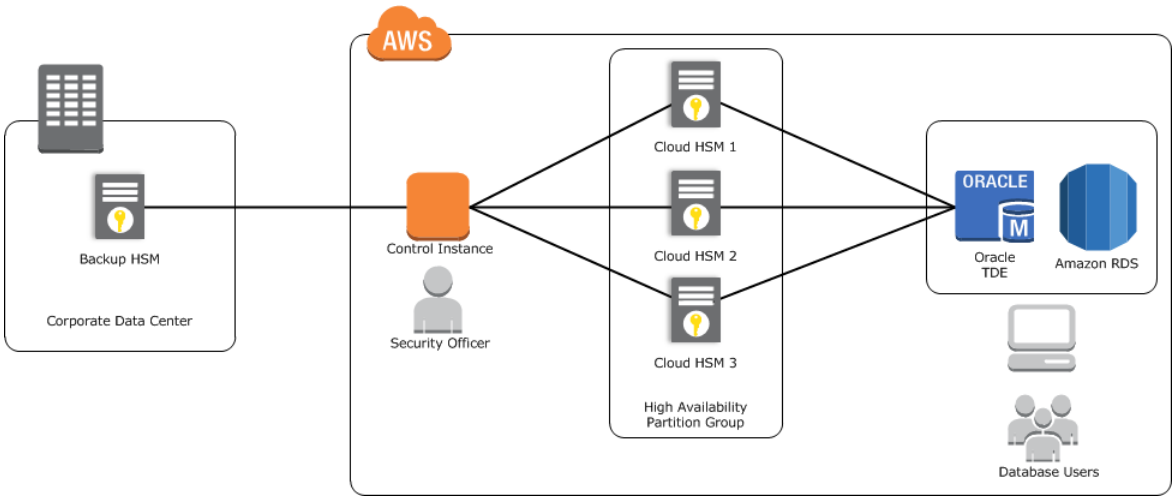
Follow the procedure in the Setting Up AWS CloudHSM section in the *AWS CloudHSM User Guide* to setup a AWS CloudHSM environment.

Installing the AWS CloudHSM Command Line Interface Tools

Follow the instructions in the Setting Up the AWS CloudHSM CLI Tools section in the *AWS CloudHSM User Guide* to install the AWS CloudHSM command line interface tools on your AWS CloudHSM control instance.

Configuring Your HSMs

The recommended configuration for using AWS CloudHSM with Amazon RDS is to use three AWS CloudHSM appliances configured into a high-availability (HA) partition group. A minimum of three HSMs are suggested for HA purposes. Even if two of your HSMs are unavailable, your keys will still be available to Amazon RDS.



Important
Initializing an HSM sets the password for the HSM security officer account (also known as the HSM administrator). Record the security officer password on your Password Worksheet (p. 273) and do not lose it. We recommend that you print out a copy of the Password Worksheet (p. 273), use it to record your AWS CloudHSM passwords, and store it in a secure place. We also recommended that you store at least one copy of this worksheet in secure off-site storage. AWS does not have the ability to recover your key material from an HSM for which you do not have the proper HSM security officer credentials.

To provision and initialize your HSMs using the AWS CloudHSM CLI tools, perform the following steps from your control instance:

1. Following the instructions in the Creating Your HSMs with the CLI section in the *AWS CloudHSM Command Line Interface Tools Reference*, provision the number of HSMs you need for your configuration. When you provision your HSMs, make note of the ARN of each HSM because you will need these to initialize your HSMs and create your high-availability partition group.
2. Following the instructions in the Initializing Your HSMs section in the *AWS CloudHSM Command Line Interface Tools Reference*, initialize each of your HSMs.

Creating Your High-Availability Partition Group

After your HSMs are initialized, create an HA partition group with the initialized HSMs. Creating an HA partition group is a three-step process. You create the HA partition group, add your HSMs to the HA partition group, and register the clients for use with the HA partition group.

To create and initialize an HA partition group

1. Following the instructions in the Create the HA Partition Group section in the *AWS CloudHSM Command Line Interface Tools Reference*, create your HA partition group. Save the HA partition group ARN returned from the **create-hapg** command for later use.

   Save the partition password on your Password Worksheet (p. 273).
2. Following the instructions in the Registering a Client with a High-Availability Partition Group section in the *AWS CloudHSM Command Line Interface Tools Reference*, create, register, and assign the clients to be used with your HA partition group.

Repeat this process to add additional partitions if necessary. One partition can support multiple Oracle databases.

Password Worksheet

Use the following worksheet to compile information for your AWS CloudHSM appliances. Print this page and use it to record your AWS CloudHSM passwords, and store it in a secure place. We also recommended that you store at least one copy of this worksheet in secure off-site storage.

Security Officer Password

This password was set when you initialized the HSM appliance.

_____

Manager Password (Optional)

This password was optionally set with the **user password manager** command on the HSM appliance.

_____

Partition Passwords

Partition Label	Partition Password	Cloning Domain

Setting Up Amazon RDS to Work with AWS CloudHSM

To use AWS CloudHSM with an Oracle DB instance using Oracle TDE, you must do the following tasks:

- Ensure that the security group associated with the Oracle DB instance allows access to the HSM port 1792.
- Create a DB subnet group that uses the same subnets as those in the VPC used by your HSMs, and then assign that DB subnet group to your Oracle DB instance.
- Set up the Amazon RDS CLI.
- Add IAM permissions for Amazon RDS to use when accessing AWS CloudHSM.
- Add the **TDE_HSM** option to the option group associated with your Oracle DB instance using the Amazon RDS CLI.
- Add two new DB instance parameters to the Oracle DB instance that will use AWS CloudHSM. The `tde-credential-arn` parameter is the Amazon Resource Number (ARN) of the high-availability (HA) partition group returned from the `create-hapg` command. The `tde-credential-password` is the partition password you used when you initialized the HA partition group.

The Amazon RDS CLI documentation can be found at Setting Up the Command Line Tools. General instructions on using the Amazon RDS CLI can be found at Amazon RDS Command Line Toolkit.

The following sections show you how to set up the Amazon RDS CLI, add the required permissions for RDS to access your HSMs, create an option group with the **TDE_HSM** option, and how to create or modify a DB instance that will use the **TDE_HSM** option.

Security Group

To allow the RDS instance to communicate with the HSM, the security group ENI assigned to the HSM appliance must authorize ingress connectivity on TCP port 1792 from the DB instance. Additionally, the Network ACL associated with the HSM's ENI must permit ingress TCP port 1792 from the RDS instance, and egress connections from the HSM to the Dynamic Port range on the RDS instance. For more information about the Dynamic TCP Port range, please see the Amazon VPC documentation.

If you used the AWS CloudFormation template to create your AWS CloudHSM environment, modify the security group that has `Allows SSH and NTLS from the public subnet` for the description. If you didn't use the AWS CloudFormation template, modify the security group associated with the ENI assigned to the HSM appliance.

DB Subnet Group

The DB subnet group that you assign to your Oracle DB instance must have the same subnets as those in the VPC used by the CloudHSM. For information about how to create a DB subnet group, see Creating a DB Subnet Group, or you can use the RDS CLI to create the DB subnet group.

Setting Up the Amazon RDS CLI

The Amazon RDS CLI can be installed on a computer running the Linux or Windows operating system and that has Java version 1.6 or higher installed.

The following steps install and configure the Amazon RDS CLI:

1. Download the Amazon RDS CLI from here. Unzip the file.
2. Set the following environment variables:

```
AWS_RDS_HOME - <The directory where the deployment files were copied to>
JAVA_HOME - <Java Installation home directory>
```

You can check that the environment variables are set correctly by running the following command for Linux or Windows:

Linux: `ls ${AWS_RDS_HOME}/bin` should list rds-describe-db-instances and the other Amazon RDS CLI commands

Windows: `dir %AWS_RDS_HOME%\bin` should list rds-describe-db-instances and the other Amazon RDS CLI commands

3. Add `${AWS_RDS_HOME}/bin` (Linux) or `%AWS_RDS_HOME%\bin` (Windows) to your path
4. Add the RDS service URL information for your AWS region to your shell configuration. For example:

```
export RDS_URL=https://rds.us-east-1.amazonaws.com
export SERVICE_SIG_NAME=rds
```

5. If you are on a Linux system, set execute permissions on all files in the bin directory using the following command:

```
chmod +x ${AWS_RDS_HOME}/bin/*
```

6. Provide the Amazon RDS CLI with your AWS user credentials. There are two ways you can provide credentials: AWS keys, or using X.509 certificates.

   If you are using AWS keys, do the following:

   a. Edit the credential file included in the zip file, ${AWS_RDS_HOME}/credential-file-path.template, to add your AWS credentials. If you are on a Linux system, limit permissions to the owner of the credential file:

   ```
   $ chmod 600 <credential file>
   ```

   b. Alternatively, you can provide the following option with every command:

   ```
   $ <RDSCLIcommand> --aws-credential-file <credential file>
   ```

   c. Or you can explicitly specify credentials on the command line: --I ACCESS_KEY --S SECRET_KEY

   If you are using X.509 certifications, do the following:

   a. Save your certificate and private keys to files: e.g. my-cert.pem and my-pk.pem.

   b. Set the following environment variables:

   ```
   EC2_CERT=<path_to_my_cert>
   EC2_PRIVATE_KEY=<path_to_my_private_key>
   ```

   c. Or you can specify the files directly on command-line for every command:

   ```
   <RDSCLIcommand> --ec2-cert-file-path=<path_to_my_cert> --ec2-private-key-
   file-path=<path_to_my_private_key>
   ```

You can test that you have set up the Amazon RDS CLI correct by running the following commands. The first command should output the usage page for all Amazon RDS commands. The second command should output information on all DB instances for the account you are using.

```
rds --help
rds-describe-db-instances --headers
```

Adding IAM Permissions for Amazon RDS to Access the AWS CloudHSM

You can use a single AWS account to work with Amazon RDS and AWS CloudHSM or you can use two separate accounts, one for Amazon RDS and one for AWS CloudHSM. This section provides information on both processes.

Topics

- Adding IAM Permissions for a Single Account for Amazon RDS to Access the AWS CloudHSM API (p. 276)
- Using Separate AWS CloudHSM and Amazon RDS Accounts for Amazon RDS to Access CloudHSM (p. 276)

Adding IAM Permissions for a Single Account for Amazon RDS to Access the AWS CloudHSM API

To create a IAM role that Amazon RDS uses to access the AWS CloudHSM API, use the following procedure. Amazon RDS checks for the presence of this IAM role when you create or modify a DB instance that uses AWS CloudHSM.

To create a IAM role for Amazon RDS to access the AWS CloudHSM API

1. Open the IAM Console at https://console.aws.amazon.com.
2. In the left navigation pane, click **Roles**.
3. Click **Create New Role**.
4. In the **Role Name** text box, type **`RDSCloudHsmAuthorization`**. Currently, you must use this name. Click **Next Step**.
5. Click **AWS Service Roles**, scroll to **Amazon RDS**, choose **Select**.
6. On the **Attach Policy** page, click **Next Step**. The correct policy is already attached to this role.
7. Review the information and then click **Create Role**.

Using Separate AWS CloudHSM and Amazon RDS Accounts for Amazon RDS to Access CloudHSM

If you want to separately manage your AWS CloudHSM and Amazon RDS resources, you can use the two services with separate accounts. To use two different accounts, you must set up each account as described in the following section.

To use two accounts, you must have the following:

- An account that is enabled for the AWS CloudHSM service and that is the owner of your hardware security module (HSM) devices. Generally, this account is your CloudHSM account, with a customer ID of HSM_ACCOUNT_ID.
- An account for Amazon RDS that you can use to create and manage a DB instance that uses Oracle TDE. Generally, this account is your DB account, with a customer ID DB_ACCOUNT_ID.

To add DB account permission to access CloudHSM resources under the CloudHSM account

1. Open the IAM Console at https://console.amazonaws.cn/.
2. Log in using your DB account.
3. In the left navigation pane, choose **Roles**.
4. Choose **Create New Role**.
5. For **Role Name**, type **`RDSCloudHsmAssumeAuthorization`**. Currently, you must use this role name for this approach to work. Choose **Next Step**.
6. Choose **AWS Service Roles**, scroll to **Amazon RDS**, choose **Select**.
7. On the **Attach Policy** page, do not attach a policy. Choose **Next Step**.
8. Review the information, and then choose **Create Role**.
9. For Roles, choose the **RDSCloudHsmAssumeAuthorization** role.
10. For Permissions, choose **Inline Policies**. Text appears that provides a link; click **click here**.
11. On the **Set Permissions** page, choose **Custom Policy**, then choose **Select**.
12. For **Policy Name**, type **`AssumeRole`**.
13. For **Policy Document**, type the following policy information:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": "*"
        }
    ]
}
```

14. Choose **Apply Policy**, and then log out of your DB account.

To revise the CloudHSM HSM account to trust permission to access CloudHSM resources under the CloudHSM account

1. Open the IAM Console at https://console.amazonaws.cn/.
2. Log in using your CloudHSM account.
3. In the left navigation pane, choose **Roles**.
4. Choose the **RDSCloudHsmAuthorization** role. This role is the one created for a single account CloudHSM-RDS.
5. Choose **Edit Trust Relationship**.
6. Add your DB account as a trusted account. The policy document should look like the following, with your DB account replacing the <*DB_ACCOUNT_ID*> placeholder:

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com",
       "AWS":[   "arn:aws:iam::$<DB_ACCOUNT_ID>$:role/RDSCloudHsmAssumeAu
thorization"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

7. Choose **Update Trust Policy**.

Creating an Amazon VPC Using the DB Account That Can Connect to Your HSM

HSM appliances are provisioned into an HSM-specific Amazon VPC. By default, only hosts inside the HSM VPC can see the HSM devices. Thus, all DB instances need to be created inside the HSM VPC or in a VPC that can be linked to the HSM VPC using VPC peering.

To use CloudHSM with an Amazon RDS DB instance in a different VPC (which you create under your DB account, as described in Creating a DB Instance in a VPC (p. 566)), you set up VPC peering from the VPC containing the DB instance to the HSM-specific VPC that contains your HSM appliances.

To set up VPC peering between the two VPCs

1. Use an existing VPC created under your DB account, or create a new VPC using your DB account. The VPC should not have any CIDR ranges that overlap with the CIDR ranges of the HSM-specific VPC.
2. Perform VPC peering between the DB VPC and the HSM VPC. For instructions, go to http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-peering.html in the *Amazon Virtual Private Cloud User Guide.*
3. Ensure that the VPC routing table is correctly associated with the VPC subnet and the VPC security group on the HSM network interface.

Note that you must configure both VPCs' routing tables so that network traffic goes to the correct VPC (from the DB VPC to the HSM VPC, and from the HSM VPC to the DB VPC). The two VPCs don't need to share the same security group, though the security groups must not prevent network traffic between the two VPCs.

Creating an Option Group with the TDE_HSM Option

The **TDE_HSM** option can be added to an existing option group just like other Oracle options, or you can create a new option group and add the **TDE_HSM** option. The following Amazon RDS CLI example creates an option group for Oracle Enterprise Edition 11.2 named *tdehsm-option-group*.

```
$ rds-create-option-group tdehsm-option-group --description "Option Group with
 TDE_HSM" --engine-name oracle-ee --major-engine-version 11.2
```

The output of the command should appear similar to the following example:

```
OPTIONGROUP  tdehsm-option-group  oracle-ee  11.2  Option Group with TDE_HSM
n
```

Once the option group has been created, you can use the following command to add the **TDE_HSM** option to the option group.

```
$ rds-add-option-to-option-group tdehsm-option-group --option-name TDE_HSM
```

The output of the command should appear similar to the following example:

```
OPTION  TDE_HSM  y  n  Oracle Advanced Security - TDE with HSM
```

Adding the AWS CloudHSM Parameters to an Oracle DB Instance

An Oracle Enterprise Edition DB instance that uses AWS CloudHSM must have two new parameters added to the DB instance. The `tde-credential-arn` and `tde-credential-password` parameters are new parameters you must include when creating a new DB instance or when modifying an existing DB instance to use AWS CloudHSM.

Creating a New Oracle DB Instance with Additional Parameters for AWS CloudHSM

When creating a new DB instance to use with AWS CloudHSM, there are several requirements:

- You must include the option group that contains the **TDE_HSM** option
- You must provide values for the `tde-credential-arn` and `tde-credential-password` parameters. The `tde-credential-arn` parameter value is the Amazon Resource Number (ARN) of the HA partition group returned from the `create-hapg` command. You can also retrieve the ARNs of all of your high-availability partition groups with the `list-hapgs` command.

  The `tde-credential-password` is the partition password you used when you initialized the HA partition group.
- The IAM Role that provides cross-service access must be created.
- You must create an Oracle Enterprise Edition DB instance.

The following command creates a new Oracle Enterprise Edition DB instance called *HsmInstance-test01* that includes the two parameters that provide AWS CloudHSM access and uses an option group called *tdehsm-option-group*.

```
$ rds-create-db-instance HsmInstance-test01
--db-instance-class <instance class>
--engine oracle-ee
--tde-credential-arn <ha partition group ARN>
--tde-credential-password <partition password>
--db-name <Oracle DB instance name>
--db-subnet-group-name <subnet group name>
--connection-timeout <connection timeout value>
--master-user-password <master user password>
--master-username <master user name>
--allocated-storage <storage value>
--option-group <TDE option group>
```

The output of the command should appear similar to the following example:

```
DBINSTANCE  hsminstance-test01  db.m1.medium  oracle-ee  40  fooooo  creating

1  ****  n  11.2.0.2.v7  bring-your-own-license  AL52UTF8  n
      VPCSECGROUP  sg-922xvc2fd  active
SUBNETGROUP  dev-test  test group  Complete  vpc-3facfe54
      SUBNET  subnet-1fd6a337  us-east-1e  Active
      SUBNET  subnet-28aeff43  us-east-1c  Active
      SUBNET  subnet-5daeff36  us-east-1b  Active
      SUBNET  subnet-2caeff47  us-east-1d  Active
      PARAMGRP  default.oracle-ee-11.2  in-sync
      OPTIONGROUP  tdehsm-option-group  pending-apply
```

Modifying an Existing DB Instance to Add Parameters for AWS CloudHSM

The following command modifies an existing Oracle Enterprise Edition DB instance and adds the `tde-credential-arn` and `tde-credential-password` parameters. Note that you must also include in the command the option group that contains the **TDE_HSM** option.

```
$ rds-modify-db-instance hsm03
--tde-credential-arn <ha partition group ARN>
--tde-credential-password <partition password>
--option-group <tde hsm option group>
--apply-immediately
```

The output of the command should appear similar to the following example:

```
DBINSTANCE  hsm03  2014-04-03T18:48:53.106Z  db.m1.medium  oracle-ee  40  fooooo
  available
hsm03.c1iibpgwvdfo.us-east-1.rds.amazonaws.com  1521  us-east-1e  1
n  11.2.0.2.v7  bring-your-own-license  AL32UTF8  n
      VPCSECGROUP  sg-922dc2fd  active
SUBNETGROUP  dev-test  test group  Complete  vpc-3faffe54
      SUBNET  subnet-1fd6a337  us-east-1e  Active
      SUBNET  subnet-28aeff43  us-east-1c  Active
      SUBNET  subnet-5daeff36  us-east-1b  Active
      SUBNET  subnet-2caeff47  us-east-1d  Active
      PARAMGRP  default.oracle-ee-11.2  in-sync
      OPTIONGROUP  tdehsm-option-group              pending-apply
      OPTIONGROUP  default:oracle-ee-11-2  pending-removal
```

Verifying the HSM Connection, the Oracle Keys in the HSM, and the TDE Key

Once you have completed all the set up steps, you can verify the HSM is working properly for TDE key storage. Connect to the Oracle DB instance using a SQL utility such as *sqlplus* on a client computer or from the EC2 control instance if it has *sqlplus* installed. For more information on connecting to an Oracle DB instance, see Connecting to a DB Instance Running the Oracle Database Engine.

> **Note**
> Before you continue, you must verify that the option group that you created for your Oracle instance returns a status of `in-sync`. You can verify this passing the DB instance identifier to the `rds-describe-db-instances` command.

Verifying the HSM Connection

You can verify the connection between an Oracle DB instance and the HSM. Connect to the Oracle DB instance and use the following command:

```
$ select * from v$encryption_wallet;
```

If the HSM connection is working, the command should return a status of *OPEN*. The output of the command will be similar to the following example:

```
WRL_TYPE
--------------------
WRL_PARAMETER
-------------------
STATUS
------------------
HSM
OPEN

1 row selected.
```

Verifying the Oracle Keys in the HSM

Once Amazon RDS starts and Oracle is running, Oracle creates two master keys on the HSM. Do the following steps to confirm the existence of the master keys in the HSM. You can run these commands from the prompt on the EC2 control instance or from the Amazon RDS Oracle DB instance.

1. Use SSH to connect to the HSM appliance. The following command

   ```
   $ ssh manager@10.0.203.58
   ```

2. Log in to the HSM as the HSM manager

   ```
   $ hsm login
   ```

3. Once you have successfully logged in, the Luna Shell prompt appears ([hostname]lunash:>). Display the contents of the HSM partition that corresponds to the Oracle DB instance using TDE. Look for two symmetric key objects that begin with "ORACLE.TDE.HSM."

   ```
   lunash:>part showContents -par <hapg_label> -password <partition_password>
   ```

   The following output is an example of the information returned from the command:

   ```
   Partition Name:  hapg_label
   Partition SN:    154749011
   Storage (Bytes): Total=102701, Used=348, Free=102353
   Number objects:  2

   Object Label:  ORACLE.TDE.HSM.MK.0699468E1DC88E4F27BF426176B94D4907
   Object Type:   Symmetric Key

   Object Label:  ORACLE.TSE.HSM.MK.0784B1918AB6C19483189B2296FAE261C70203
   ```

```
Object Type:   Symmetric Key

Command Result : 0 (Success)
```

Verifying the TDE Key

The final step to verifying that the TDE key is correctly stored in the HSM is to create an encrypted tablespace. The following commands creates an encrypted tablespace and shows that it is encrypted.

```
SQL> create tablespace encrypted_ts
datafile size 50M encryption using 'AES128'
default storage (encrypt)
/
SQL> select tablespace_Name, encrypted from dba_tablespaces where encrypted='YES'
```

The following sample output shows that the tablespace was encrypted:

```
TABLESPACE_NAME                ENC
------------------------------ ---
ENCRYPTED_TS                   YES
```

Restoring Encrypted DB Instances

To restore an encrypted Oracle DB instance, you can use your existing AWS CloudHSM HA partition group or create a new HA partition group and copy the contents from the original partition group to the new partition group. Please update the SafeNet client on your HSM control instance if you would like to use your existing HA partition group. Then use the **rds-restore-db-instance-from-db-snapshot** command to restore the DB instance.

To restore the instance, perform the following procedure:

1. On your AWS CloudHSM control instance, create a new HA partition group as shown in Creating Your High-Availability Partition Group (p. 272). When you create the new HA partition group, you must specify the same partition password as the original HA partition group. Make a note of the ARN of the new HA partition group, which you will need in the next two steps.
2. On your AWS CloudHSM control instance, clone the contents of the existing HA partition group to the new HA partition group with the **clone-hapg** command.

   ```
   $ cloudhsm clone-hapg --conf_file ~/cloudhsm.conf
   --src-hapg-arn <src_arn>
   --dest-hapg-arn <dest_arn>
   --client-arn <client_arn>
   --partition-password <partition_password>
   ```

   The parameters are as follows:

   <src_arn>
   : The identifier of the existing HA partition group.

   <dest_arn>
   : The identifier of the new HA partition group created in the previous step.

<client_arn>
The identifier of the HSM client.

<partition_password>
The password for the member partitions. Both HA partition groups must have the same partition password.

3. Use the **rds-restore-db-instance-from-db-snapshot** command to restore the DB instance. In the restore command, pass the ARN of the new HA partition group in the **tde-credential-arn** parameter, and the partition password for the HA partition group in the **tde-credential-password** parameter.

Managing a Multi-AZ Failover

You do not need to set up a AWS CloudHSM HA partition group for your standby DB instance if you are using a Multi-AZ deployment. In fact, the details of a failover are handled automatically for you. During a failover, the standby instance becomes the new primary instance and the HSM continues to work with the new primary instance.

Appendix: Oracle Character Sets Supported in Amazon RDS

The following table lists the Oracle database character sets that are supported in Amazon RDS. You can use a value from this page with the `--character-set` parameter of the **rds-create-db-instance** command or with the `CharacterSetName` parameter of the `CreateDBInstance` API action.

Setting the NLS_LANG environment parameter is the simplest way to specify locale behavior for Oracle software. This parameter sets the language and territory used by the client application and the database server. It also indicates the client's character set, which corresponds to the character set for data entered or displayed by a client application. Amazon RDS lets you set the character set when you create a DB instance. For more information on the NLS_LANG and character sets, see What is a Character set or Code Page? in the Oracle documentation.

Value	Description
AL32UTF8	Unicode 5.0 UTF-8 Universal character set (default)
AR8ISO8859P6	ISO 8859-6 Latin/Arabic
AR8MSWIN1256	Microsoft Windows Code Page 1256 8-bit Latin/Arabic
BLT8ISO8859P13	ISO 8859-13 Baltic
BLT8MSWIN1257	Microsoft Windows Code Page 1257 8-bit Baltic
CL8ISO8859P5	ISO 88559-5 Latin/Cyrillic
CL8MSWIN1251	Microsoft Windows Code Page 1251 8-bit Latin/Cyrillic
EE8ISO8859P2	ISO 8859-2 East European
EL8ISO8859P7	ISO 8859-7 Latin/Greek
EE8MSWIN1250	Microsoft Windows Code Page 1250 8-bit East European
EL8MSWIN1253	Microsoft Windows Code Page 1253 8-bit Latin/Greek
IW8ISO8859P8	ISO 8859-8 Latin/Hebrew
IW8MSWIN1255	Microsoft Windows Code Page 1255 8-bit Latin/Hebrew
JA16EUC	EUC 24-bit Japanese
JA16EUCTILDE	Same as JA16EUC except for mapping of wave dash and tilde to and from Unicode
JA16SJIS	Shift-JIS 16-bit Japanese
JA16SJISTILDE	Same as JA16SJIS except for mapping of wave dash and tilde to and from Unicode
KO16MSWIN949	Microsoft Windows Code Page 949 Korean

Value	Description
NE8ISO8859P10	ISO 8859-10 North European
NEE8ISO8859P4	ISO 8859-4 North and Northeast European
TH8TISASCII	Thai Industrial Standard 620-2533-ASCII 8-bit
TR8MSWIN1254	Microsoft Windows Code Page 1254 8-bit Turkish
US7ASCII	ASCII 7-bit American
UTF8	Unicode 3.0 UTF-8 Universal character set, CESU-8 compliant
VN8MSWIN1258	Microsoft Windows Code Page 1258 8-bit Vietnamese
WE8ISO8859P1	Western European 8-bit ISO 8859 Part 1
WE8ISO8859P15	ISO 8859-15 West European
WE8ISO8859P9	ISO 8859-9 West European and Turkish
WE8MSWIN1252	Microsoft Windows Code Page 1252 8-bit West European
ZHS16GBK	GBK 16-bit Simplified Chinese
ZHT16HKSCS	Microsoft Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001. Character set conversion is based on Unicode 3.0.
ZHT16MSWIN950	Microsoft Windows Code Page 950 Traditional Chinese
ZHT32EUC	EUC 32-bit Traditional Chinese

Appendix: Oracle Database Engine Release Notes

This section provides information about what's new and what patch sets are included in each Amazon RDS release for the Oracle DB engine.

Amazon RDS incorporates Oracle Database bug fixes from Oracle via their quarterly Patch Set Updates (PSU). We do not support applying one-off patches to individual DB instances; you can be confident that your DB instance is running a stable, common version of the database software that has been regression tested by both Oracle and Amazon.

Topics

- Database Engine Version: 11.2.0.2.v3 (p. 287)
- Database Engine Version: 11.2.0.2.v4 or 11.2.0.2.v5 (p. 287)
- Database Engine Version: 11.2.0.2.v6 (p. 288)
- Database Engine Version: 11.2.0.2.v7 (p. 289)
- Database Engine Version: 11.2.0.3.v1 (p. 290)
- Database Engine Version: 11.2.0.3.v2 (p. 291)
- Database Engine Version: 11.2.0.3.v3 (p. 293)
- Database Engine Version: 11.2.0.4.v1 (p. 294)
- Database Engine Version: 11.2.0.4.v2 (Deprecated) (p. 295)
- Database Engine Version: 11.2.0.4.v3 (p. 295)
- Database Engine Version: 11.2.0.4.v4 (p. 296)
- Database Engine Version: 12.1.0.1.v1 (p. 297)
- Database Engine Version: 12.1.0.1.v2 (p. 298)
- Database Engine Version: 12.1.0.2.v1 (p. 299)

The following table shows what Oracle PSUs are applied to the Oracle versions in Amazon RDS:

PSU	Version 11.2.0.2	Version 11.2.0.3	Version 11.2.0.4	Version 12.1.0.1	Version 12.1.0.2
PSU July 2011	11.2.0.2.v3				
PSU July 2012	11.2.0.2.v4 and 11.2.0.2.v5				
PSU October 2012	11.2.0.2.v6				
PSU April 2013	11.2.0.2.v7				
PSU July 2013		11.2.0.3.v1			
PSU January 2014			11.2.0.4.v1		
PSU July 2014			11.2.0.4.v2 (Deprecated)		
PSU October 2014		11.2.0.3.v2	11.2.0.4.v3		
PSU January 2015				12.1.0.1.v1	

PSU	Version 11.2.0.2	Version 11.2.0.3	Version 11.2.0.4	Version 12.1.0.1	Version 12.1.0.2
PSU April 2015		11.2.0.3.v3	11.2.0.4.v4	12.1.0.1.v2	12.1.0.2.v1

Oracle has determined that patching for the following versions will end (support Doc 742060.1):

- Oracle version 11.2.0.2 – Patching ended October 2013
- Oracle version 11.2.0.3 – Patching is set to end in July 2015

Database Engine Version: 11.2.0.2.v3

What's New in Version 11.2.0.2.v3

This version includes Oracle PSU 11.2.0.2.3.

Baseline: Oracle Database Patch Set Update (PSU) 11.2.0.2.3

Bugs fixed: 10151017, 10158965, 11724916, 10190642, 12586486, 12586487, 10129643, 12586488, 12586489, 10018789, 9744252, 10248523, 9956713, 10356513, 9715581, 9770451, 10378005, 10170431, 10425676, 10222719, 10126094, 9591812, 10127360, 10132870, 10094201, 9443361, 10193846, 11664046, 11069199, 10324294, 10245086, 12586490, 10205230, 12586491, 10052141, 12586492, 12586493, 12586494, 10142788, 11818335, 11830776, 12586495, 9905049, 11830777, 12586496, 11830778, 6892311, 10040921, 10077191, 10358019, 12431716, 10219576, 10258337, 11707699, 10264680, 10209232, 11651810, 10102506, 11067567, 9881076, 10278372, 10040531, 10621169, 10155605, 10082277, 10356782, 10218814, 9078442, 9788588, 10157249, 9735237, 10317487, 12326246, 11707302, 10310299, 10636231, 10230571, 11065646, 12419321, 10368698, 10079168, 10013431, 10228151, 10233732, 10324526, 8223165, 10238786, 10217802, 10061015, 9953542, 9572787, 10052956, 10080579, 11699057, 12620422, 10332111, 10227288, 10329146, 10332589, 10110863, 10073683, 9869401, 10019218, 10229719, 11664719, 9539440, 10373381, 9735282, 9748749, 11724984, 10022980, 10411618, 11800854, 12419331, 11674485, 10187168, 6523037, 10648873, 9724970, 10053725, 10084145, 10367188, 11800170, 11695285, 10157402, 9651350, 10299224

Database Engine Version: 11.2.0.2.v4 or 11.2.0.2.v5

What's New in Version 11.2.0.2.v4 or 11.2.0.2.v5

This version includes Oracle PSU 11.2.0.2.7 and adds support for importing data using Oracle Data Pump.

Baseline: Oracle Database Patch Set Update (PSU) 11.2.0.2.7

Bugs fixed: 10249791, 11877623, 12569737, 14038791, 10026601, 12378147, 10115630, 11814891, 14127510, 10412247, 13923804, 12656535, 9709292, 10220033, 10092858, 12391602, 12323180, 10142857, 10620808, 12579349, 12337012, 12879027, 11811073, 11064851, 13001379, 9903826, 11738259, 14107384, 10207092, 14107385, 11882425, 9858539, 14107386, 14107387, 10633840, 14107388, 10419629, 14107389, 11708510, 10131867, 14040433, 11063191, 13916709, 12880299, 11872103, 12595730, 11056082, 12596444, 13099577, 13632725, 10031806, 13769501, 13769502, 13769503, 13769504, 9744252, 13769505, 9956713, 13769506, 13769507, 9972680, 13769508, 13769509, 11853815, 10635701, 9591812, 10127360, 11723722, 9443361, 12846268, 12846269, 9707965, 10245086, 9401552, 10039731, 11689702, 13769510, 12366627, 10077191, 9829397, 11785938, 10258337, 10264680, 10094823, 10209232, 10284570, 8672862, 9672816, 12830339, 9881076, 10621169, 10048701, 12569482, 9078442, 11057263, 10322959, 12780098, 12976376,

12340939, 11788856, 8223165, 10264696, 10142909, 11800959, 13476583, 10052956, 10285022, 10329146, 10332589, 9895207, 9869401, 12828071, 9285259, 10229719, 11724984, 10411618, 11670161, 9724970, 10113990, 10312847, 11893621, 10200390, 10084145, 10367188, 10285394, 10190642, 12586486, 12586487, 10129643, 12586488, 12917230, 12586489, 11866952, 10232083, 9715581, 10302581, 11690639, 12423475, 11889177, 10126094, 10396041, 10269503, 9970255, 9436324, 12400751, 12589039, 11785390, 12586490, 12586491, 12586492, 9795214, 12586493, 10142788, 12586494, 12586495, 9905049, 12586496, 11674898, 10419984, 6892311, 11815753, 10358019, 12431716, 9906422, 10422126, 13343244, 11937253, 9965655, 11890804, 11651810, 9382956, 11067567, 11716621, 10126822, 9869287, 9375300, 10155605, 10356782, 10326338, 10165083, 10051315, 13696224, 10218814, 13554409, 11076894, 10278773, 11707302, 10230571, 12419321, 9966609, 12633340, 12546006, 10137324, 11894889, 10061015, 9572787, 10284838, 10073683, 12639234, 9578670, 9748749, 10022980, 10237773, 10089333, 12419331, 11674485, 12685431, 10187168, 10648873, 10158965, 11061775, 12635537, 9746210, 10204358, 10356513, 10378005, 10170431, 12639177, 10222719, 10384285, 10035737, 12345717, 9873405, 11069199, 12670165, 10159846, 13257247, 10205230, 10052141, 11818335, 12371955, 12655433, 10040921, 11827088, 10219576, 12408350, 13343424, 11707699, 12370722, 11695333, 11841309, 11924400, 12737666, 12797765, 10281887, 10278372, 10013177, 13503598, 12543639, 10157249, 12531263, 9735237, 10317487, 10219583, 9727147, 10310299, 10636231, 11065646, 10055063, 10368698, 10079168, 11695416, 10233732, 10314582, 9953542, 10080579, 11699057, 12620422, 10427260, 11666137, 10110863, 10363186, 10417716, 10019218, 10388660, 12748240, 9539440, 10373381, 10239480, 10158493, 11842991, 10399808, 10417216, 11695285, 11800170, 10157402, 9651350, 10299224, 10151017, 11724916, 9564886, 9847634, 10018789, 10248523, 11694127, 10630870, 9770451, 10425676, 9683047, 10180307, 9835264, 10132870, 10094201, 10193846, 11664046, 10324294, 9414040, 9819805, 11830776, 11830777, 11830778, 11683713, 10200404, 10102506, 12827726, 11733179, 10229886, 10040531, 10082277, 9788588, 12326246, 12397410, 10622001, 13468884, 13386082, 10040035, 12539000, 11867127, 9842573, 9771278, 10013431, 10228151, 10324526, 12417369, 10238786, 10217802, 10332111, 10227288, 10623249, 9943960, 10021022, 9824435, 11664719, 12950644, 9735282, 11800854, 10097711, 11858315, 6523037, 10053725, 8685446

Database Engine Version: 11.2.0.2.v6

What's New in Version 11.2.0.2.v6

This version includes Oracle PSU 11.2.0.2.8.

Baseline: Oracle Database Patch Set Update (PSU) 11.2.0.2.8

Bugs fixed: 13250244, 13737746, 11063821, 12409916, 14461356, 14461357, 11878443, 14461358, 14683459, 14275621, 14467061, 10114837, 12649442, 10207551, 12794305, 14473913, 10171273, 10373013, 10210507, 11883472, 13080778, 10172453, 14624146, 14613900, 10213073, 9373370, 9478199, 9877980, 10021111, 10228393, 12899768, 12713993, 9470768, 14390377, 10140809, 12894807, 11686968, 12374212, 12764337, 12326708, 9956835, 11734067, 7312717, 11775474, 12834027, 13326736, 9952554, 10249791, 11877623, 12569737, 14038791, 10026601, 12378147, 10115630, 11814891, 14127510, 10412247, 13923804, 12656535, 9709292, 10220033, 10092858, 12391602, 12323180, 10142857, 10620808, 12579349, 12337012, 12879027, 11811073, 11064851, 13001379, 9903826, 11738259, 14107384, 10207092, 14107385, 11882425, 9858539, 14107386, 14107387, 10633840, 14107388, 10419629, 14107389, 11708510, 10131867, 14040433, 11063191, 13916709, 12880299, 11872103, 12595730, 11056082, 12596444, 13099577, 13632725, 10031806, 13769501, 13769502, 13769503, 13769504, 9744252, 13769505, 9956713, 13769506, 13769507, 9972680, 13769508, 13769509, 11853815, 10635701, 9591812, 10127360, 11723722, 9443361, 12846268, 12846269, 9707965, 10245086, 9401552, 10039731, 11689702, 13769510, 12366627, 10077191, 9829397, 11785938, 10258337, 10264680, 10094823, 10209232, 10284570, 8672862, 9672816, 12830339, 9881076, 10621169, 10048701, 12569482, 9078442, 11057263, 10322959, 12780098, 12976376, 12340939, 11788856, 8223165, 10264696, 10142909, 11800959, 13476583, 10052956, 10285022, 10329146, 10332589, 9895207, 9869401, 12828071, 9285259, 10229719, 11724984, 10411618, 11670161, 9724970, 10113990, 10312847, 11893621, 10200390, 10084145,

10367188, 10285394, 10190642, 12586486, 12586487, 10129643, 12586488, 12917230, 12586489, 11866952, 10232083, 9715581, 10302581, 11690639, 12423475, 11889177, 10126094, 10396041, 10269503, 9970255, 9436324, 12400751, 12589039, 11785390, 12586490, 12586491, 12586492, 9795214, 12586493, 10142788, 12586494, 12586495, 9905049, 12586496, 11674898, 10419984, 6892311, 11815753, 10358019, 12431716, 9906422, 10422126, 13343244, 11937253, 9965655, 11890804, 11651810, 9382956, 11067567, 11716621, 10126822, 9869287, 9375300, 10155605, 10356782, 10326338, 10165083, 10051315, 13696224, 10218814, 13554409, 11076894, 10278773, 11707302, 10230571, 12419321, 9966609, 12633340, 12546006, 10137324, 11894889, 10061015, 9572787, 10284838, 10073683, 12639234, 9578670, 9748749, 10022980, 10237773, 10089333, 12419331, 11674485, 12685431, 10187168, 10648873, 10158965, 11061775, 12635537, 9746210, 10204358, 10356513, 10378005, 10170431, 12639177, 10222719, 10384285, 10035737, 12345717, 9873405, 11069199, 12670165, 10159846, 13257247, 10205230, 10052141, 11818335, 12371955, 12655433, 10040921, 11827088, 10219576, 12408350, 13343424, 11707699, 12370722, 11695333, 11841309, 11924400, 12737666, 12797765, 10281887, 10278372, 10013177, 13503598, 12543639, 10157249, 12531263, 9735237, 10317487, 10219583, 9727147, 10310299, 10636231, 11065646, 10055063, 10368698, 10079168, 11695416, 10233732, 10314582, 9953542, 10080579, 11699057, 12620422, 10427260, 11666137, 10110863, 10363186, 10417716, 10019218, 10388660, 12748240, 9539440, 10373381, 10239480, 10158493, 11842991, 10399808, 10417216, 11695285, 11800170, 10157402, 9651350, 10299224, 10151017, 11724916, 9564886, 9847634, 10018789, 10248523, 11694127, 10630870, 9770451, 10425676, 9683047, 10180307, 9835264, 10132870, 10094201, 10193846, 11664046, 10324294, 9414040, 9819805, 11830776, 11830777, 11830778, 11683713, 10200404, 10102506, 12827726, 11733179, 10229886, 10040531, 10082277, 9788588, 12326246, 12397410, 10622001, 13468884, 13386082, 10040035, 12539000, 11867127, 9842573, 9771278, 10013431, 10228151, 10324526, 12417369, 10238786, 10217802, 10332111, 10227288, 10623249, 9943960, 10021022, 9824435, 11664719, 12950644, 9735282, 11800854, 10097711, 11858315, 6523037, 10053725, 8685446

Database Engine Version: 11.2.0.2.v7

What's New in Version 11.2.0.2.v7

This version adds support for the following:

- Retaining Archived Redo Logs (for version 11.2.0.2.v7 and later) (p. 248)
- Oracle PSU 11.20.2.10

Baseline: Oracle Database Patch Set Update (PSU) 11.2.0.2.10 (April 2013)

Bugs fixed: 16344871, 9671271, 16294412, 14841558, 12579446, 16056267, 10435074, 14273397, 12428791, 12314102, 10138589, 14841812, 12842402, 16303117, 10372924, 12539487, 12594032, 13377816, 16303116, 16175381, 14220725, 13561951, 9868876, 9913542, 16303114, 10362871, 9801919, 12755116, 13524899, 16303115, 10350832, 16303118, 12582664, 13596521, 14459552, 13810393, 13147164, 15896431, 10247152, 14076523, 10395345, 14023636, 13467683, 11706168, 15896427, 14263073, 9926929, 10190172, 11715084, 15896432, 9896536, 15896428, 15896429, 14841437, 12420002, 14262913, 13399435, 10396874, 8547978, 14727315, 15896434, 14546575, 9860769, 14258925, 15896433, 14546638, 11834448, 14741727, 14546673, 12845115, 15896430, 12595561, 13550185, 14263036, 9912965, 14205448, 15896435, 14035825, 12848798, 11856395, 10175192, 14469008, 12313857, 9233544, 9681133, 13250244, 13737746, 11063821, 12409916, 14461356, 14461357, 11878443, 14461358, 14683459, 14275621, 14467061, 10114837, 12649442, 10207551, 12794305, 14473913, 10171273, 10373013, 10210507, 11883472, 13080778, 10172453, 14624146, 14613900, 10213073, 9373370, 9478199, 9877980, 10021111, 10228393, 12899768, 12713993, 9470768, 14390377, 10140809, 12894807, 11686968, 12374212, 12764337, 12326708, 9956835, 11734067, 7312717, 11775474, 12834027, 13326736, 9952554, 10249791, 11877623, 12569737, 14038791, 10026601, 12378147, 10115630, 11814891, 14127510, 10412247, 13923804,

12656535, 9709292, 10220033, 10092858, 12391602, 12323180, 10142857, 10620808, 12579349, 12337012, 12879027, 11811073, 11064851, 13001379, 9903826, 11738259, 14107384, 10207092, 14107385, 11882425, 9858539, 14107386, 14107387, 10633840, 14107388, 10419629, 14107389, 11708510, 10131867, 14040433, 11063191, 13916709, 12880299, 11872103, 12595730, 11056082, 12596444, 13099577, 13632725, 10031806, 13769501, 13769502, 13769503, 13769504, 9744252, 13769505, 9956713, 13769506, 13769507, 9972680, 13769508, 13769509, 11853815, 10635701, 9591812, 10127360, 11723722, 9443361, 12846268, 12846269, 9707965, 10245086, 9401552, 10039731, 11689702, 13769510, 12366627, 10077191, 9829397, 11785938, 10258337, 10264680, 10094823, 10209232, 10284570, 8672862, 9672816, 12830339, 9881076, 10621169, 10048701, 12569482, 9078442, 11057263, 10322959, 12780098, 12976376, 12340939, 11788856, 8223165, 10264696, 10142909, 11800959, 13476583, 10052956, 10285022, 10329146, 10332589, 9895207, 9869401, 12828071, 9285259, 10229719, 11724984, 10411618, 11670161, 9724970, 10113990, 10312847, 11893621, 10200390, 10084145, 10367188, 10285394, 10190642, 12586486, 12586487, 10129643, 12586488, 12917230, 12586489, 11866952, 10232083, 9715581, 10302581, 11690639, 12423475, 11889177, 10126094, 10396041, 10269503, 9970255, 9436324, 12400751, 12589039, 11785390, 12586490, 12586491, 12586492, 9795214, 12586493, 10142788, 12586494, 12586495, 9905049, 12586496, 11674898, 10419984, 6892311, 11815753, 10358019, 12431716, 9906422, 10422126, 13343244, 11937253, 9965655, 11890804, 11651810, 9382956, 11067567, 11716621, 10126822, 9869287, 9375300, 10155605, 10356782, 10326338, 10165083, 10051315, 13696224, 10218814, 13554409, 11076894, 10278773, 11707302, 10230571, 12419321, 9966609, 12633340, 12546006, 10137324, 11894889, 10061015, 9572787, 10284838, 10073683, 12639234, 9578670, 9748749, 10022980, 10237773, 10089333, 12419331, 11674485, 12685431, 10187168, 10648873, 10158965, 11061775, 12635537, 9746210, 10204358, 10356513, 10378005, 10170431, 12639177, 10222719, 10384285, 10035737, 12345717, 9873405, 11069199, 12670165, 10159846, 13257247, 10205230, 10052141, 11818335, 12371955, 12655433, 10040921, 11827088, 10219576, 12408350, 13343424, 11707699, 12370722, 11695333, 11841309, 11924400, 12737666, 12797765, 10281887, 10278372, 10013177, 13503598, 12543639, 10157249, 12531263, 9735237, 10317487, 10219583, 9727147, 10310299, 10636231, 11065646, 10055063, 10368698, 10079168, 11695416, 10233732, 10314582, 9953542, 10080579, 11699057, 12620422, 10427260, 11666137, 10110863, 10363186, 10417716, 10019218, 10388660, 12748240, 9539440, 10373381, 10239480, 10158493, 11842991, 10399808, 10417216, 11695285, 11800170, 10157402, 9651350, 10299224, 10151017, 11724916, 9564886, 9847634, 10018789, 10248523, 11694127, 10630870, 9770451, 10425676, 9683047, 10180307, 9835264, 10132870, 10094201, 10193846, 11664046, 10324294, 9414040, 9819805, 11830776, 11830777, 11830778, 11683713, 10200404, 10102506, 12827726, 11733179, 10229886, 10040531, 10082277, 9788588, 12326246, 12397410, 10622001, 13468884, 13386082, 10040035, 12539000, 11867127, 9842573, 9771278, 10013431, 10228151, 10324526, 12417369, 10238786, 10217802, 10332111, 10227288, 10623249, 9943960, 10021022, 9824435, 11664719, 12950644, 9735282, 11800854, 10097711, 11858315, 6523037, 10053725, 8685446

Database Engine Version: 11.2.0.3.v1

What's New in Version 11.2.0.3.v1

This version adds support for the following:

- Disconnecting a Session (for version 11.2.0.3.v1 and later) (p. 243)
- Renaming the Global Name (for version 11.2.0.3.v1 and later) (p. 243)
- Setting Force Logging (for version 11.2.0.3.v1 and later) (p. 248)
- Setting Supplemental Logging (for version 11.2.0.3.v1 and later) (p. 248)
- Setting Distributed Recovery (for version 11.2.0.3.v1 and later) (p. 250)
- Listing and Reading Files in a DB Instance Directory (for version 11.2.0.3.v1 and later) (p. 252)
- Oracle PSU 11.2.0.3.7

Baseline: Oracle Database Patch Set Update (PSU) 11.2.0.3.7 (July2013)

Bugs fixed: 13593999, 13566938, 10350832, 14138130, 12919564, 13561951, 13624984, 13588248, 13080778, 13914613, 13804294, 14258925, 12873183, 13645875, 14472647, 12880299, 14664355, 12998795, 14409183, 13719081, 14469008, 13492735, 14263036, 12857027, 13496884, 13015379, 14263073, 13742433, 13732226, 16314469, 16368108, 12905058, 6690853, 13742434, 12849688, 12950644, 13742435, 13464002, 13063120, 13534412, 12879027, 13958038, 14613900, 12585543, 13790109, 12535346, 16382448, 12588744, 11877623, 12395918, 13814739, 13786142, 12847466, 13649031, 13855490, 13981051, 12582664, 12797765, 14262913, 12923168, 16279401, 12912137, 13612575, 13384182, 13466801, 13484963, 14207163, 13724193, 13772618, 11063191, 16694777, 13070939, 12797420, 15869211, 13041324, 16279211, 16314467, 16314468, 12976376, 11708510, 13680405, 13742437, 13026410, 14589750, 13737746, 13742438, 14644185, 15841373, 13326736, 13596521, 14398795, 13579992, 13001379, 16344871, 13099577, 9873405, 13742436, 14275605, 9858539, 14841812, 11715084, 16231699, 14040433, 9703627, 12662040, 12617123, 16530565, 14207317, 12845115, 12764337, 13354082, 14459552, 13397104, 13913630, 12964067, 12983611, 13550185, 12780983, 13810393, 12583611, 14546575, 15862016, 13476583, 13489024, 11840910, 13903046, 15862017, 13572659, 16294378, 13718279, 14088346, 13657605, 13448206, 16314466, 14480676, 13419660, 13632717, 14668670, 14063281, 14110275, 13430938, 13467683, 13420224, 13812031, 14548763, 16299830, 12646784, 14512189, 12755116, 14035825, 13616375, 13427062, 12861463, 12834027, 15862021, 13632809, 13377816, 13036331, 14727310, 16619892, 13685544, 13499128, 15862018, 13584130, 16175381, 12829021, 15862019, 12794305, 14546673, 12791981, 13561750, 13503598, 13787482, 10133521, 12718090, 13848402, 13399435, 14023636, 9095696, 13860201, 12401111, 13257247, 13362079, 14176879, 12917230, 16014985, 13923374, 14220725, 13524899, 14480675, 16306019, 13559697, 12974860, 9706792, 12940620, 14480674, 13916709, 13098318, 14076523, 13773133, 15905421, 16794244, 13340388, 12731940, 13528551, 13366202, 12894807, 13343438, 13454210, 12748240, 14205448, 13385346, 14127231, 15853081, 14273397, 14467061, 12971775, 13923995, 14571027, 13582702, 13907462, 10242202, 13493847, 13857111, 13035804, 13544396, 16382353, 8547978, 14226599, 16794241, 14062795, 13035360, 12925089, 12693626, 13332439, 14038787, 11071989, 14062796, 16794243, 12913474, 14841409, 14390252, 16314470, 13370330, 13059165, 14062797, 14062794, 12959852, 12345082, 13358781, 12960925, 16703112, 9659614, 14546638, 13699124, 13936424, 14301592, 16794240, 13338048, 12938841, 12658411, 12620823, 12656535, 14062793, 12678920, 13038684, 14062792, 13807411, 16742095, 16794238, 15862022, 12594032, 13250244, 12612118, 9761357, 14053457, 13742464, 14052474, 13911821, 13457582, 7509451, 13527323, 13791364, 15862020, 13910420, 12780098, 13502183, 13696216, 13705338, 10263668, 14841558, 16794242, 15862023, 16056266, 16794239, 15862024, 13554409, 13645917, 13103913, 12772404, 13011409, 14063280, 13328193, 16799735

Database Engine Version: 11.2.0.3.v2

What's New in Version 11.2.0.3.v2

This version adds support for the following:

- Oracle Database Patch Set Update (PSU) 11.2.0.3.12 (patch 19121548, released in October 2014).
- Latest DST file (DSTv23 – patch 19396455, released in October 2014). This patch is incorporated by default in new instances only.
- Added Database Patch 19695885 – Oracle GoldenGate Integrated Extract for 11.2.0.3.12.
- Upgrade paths available: You can upgrade from 11.2.0.3.v2 to later versions of 11.2.0.3 as they become available. You can also upgrade from 11.2.0.3.v2 to 11.2.0.4.v3 or later versions of 11.2.0.4 as they become available.

Baseline: Oracle Database Patch Set Update (PSU) 11.2.0.3.12 (October2014)

Bugs fixed: 19396455, 18759211, 17432124, 16799735, 14744263, 14175146, 13652437, 16238044, 13516727, 13328193, 14050233, 13593999, 10350832, 19433746, 14138130, 12919564, 14198511, 13561951 13588248, 13080778, 13804294, 16710324, 18031683, 12873183, 16992075 14193240, 14472647, 12880299, 14799269, 13369579, 13840704, 14409183 13492735, 13496884, 12857027, 14263036, 13834436, 16038929, 13015379 14263073, 17748833, 16563678, 13732226, 13866822, 13742434, 13944971 12950644, 12899768, 17748831, 16929165, 16272008, 13063120, 13958038 14613900, 13503204, 13972394, 11877623, 13072654, 17088068, 12395918 16710753, 13429702, 13814739, 17343514, 13649031, 10256843, 13981051 15981698, 13901201, 12797765, 17333200, 19211724, 12923168, 16761566 13384182, 16279401, 13466801, 15996344, 14207163, 18673304, 13596581 13724193, 11063191, 13642044, 12940637, 18641419, 12595606, 9163477 15931756, 14052871, 18262334, 13945708, 12797420, 14123213, 13041324 12865902, 15869211, 14003090, 16314468, 16019955, 11708510, 17865671 14637368, 13026410, 13737746, 13742438, 15841373, 16347904, 15910002 16088176, 19517437, 16362358, 16505333, 14398795, 14182835, 13579992 16344871, 10182005, 10400244, 13742436, 14275605, 9858539, 14841812 16338983, 9703627, 13483354, 14393728, 14207317, 17165204, 12764337 16902043, 14459552, 14191508, 14588746, 12964067, 12780983, 12583611 14383007, 14546575, 13476583, 15862016, 13489024, 12985237, 17748830 19554106, 14088346, 13448206, 19458377, 16314466, 13419660, 18139695 12591399, 14110275, 13430938, 13467683, 17767676, 14548763, 19638161 13424216, 12834027, 13632809, 13853126, 13377816, 13036331, 14727310 9812682, 12320556, 16747736, 13584130, 16175381, 17468141, 12829021 14138823, 15862019, 12794305, 14546673, 12791981, 13503598, 13787482 10133521, 12744759, 13399435, 19433747, 14762511, 13553883, 14023636 9095696, 12977562, 14343501, 13860201, 13257247, 14176879, 13783957 16014985, 12312133, 14480675, 13146182, 16306019, 13559697, 12974860 9706792, 12940620, 13098318, 15883525, 13773133, 16794244, 13340388 13528551, 13366202, 12894807, 13259364, 12747437, 13454210, 12748240 13385346, 15987992, 13923995, 16101465, 14571027, 13582702, 12784406 13907462, 13493847, 13035804, 13857111, 16710363, 13544396, 10110625 14128555, 12813641, 8547978, 14226599, 17478415, 17050888, 17333197 9397635, 14007968, 13912931, 12693626, 12925089, 14189694, 17761775 12815057, 16721594, 13332439, 14038787, 11071989, 12596444, 14207902 14062796, 12913474, 14390252, 13370330, 16314470, 14062794, 13358781 12960925, 17333202, 9659614, 14546638, 13699124, 13936424, 19433745 9797851, 16794240, 14301592, 13338048, 12938841, 12620823, 12656535 12678920, 13719292, 14488943, 14062792, 16850197, 14791477, 13807411 16794238, 13250244, 12594032, 15862022, 15826962, 14098509, 12612118 9761357, 18096714, 14053457, 13918644, 13527323, 10625145, 12797620 18173595, 19289642, 15862020, 13910420, 12780098, 13696216, 14774091 14841558, 10263668, 13849733, 16794242, 16944698, 15862023, 16056266 13834065, 13853654, 14351566, 13723052, 18173593, 14063280, 13011409 13566938, 13737888, 13624984, 16024441, 17333199, 13914613, 17540582 14258925, 14222403, 14755945, 13645875, 12571991, 13839641, 14664355 12998795, 13719081, 14469008, 13361350, 14188650, 17019974, 13742433 14508968, 16314469, 16368108, 12905058, 6690853, 13647945, 16212405 12849688, 13742435, 13464002, 18681866, 12879027, 13534412, 18522512 12585543, 12747740, 12535346, 13878246, 13790109, 16382448, 12588744 13916549, 13786142, 12847466, 13855490, 13551402, 12582664, 13871316 14657740, 14262913, 17332800, 14558880, 14695377, 12912137, 13612575 12387467, 13484963, 14163397, 17437634, 13772618, 16694777, 13070939 15994107, 13605839, 14369664, 12391034, 12588237, 16279211, 16314467 12945879, 15901852, 12976376, 7276499, 12755231, 13680405, 13742437 14589750, 14318397, 11868640, 14644185, 13326736, 13596521, 13001379 12898558, 17752121, 13099577, 13911711, 9873405, 18673325, 16372203 16344758, 11715084, 9547706, 16231699, 14040433, 12662040, 12617123 14406648, 17748832, 16530565, 12845115, 16844086, 13354082, 17748834 13794550, 13397104, 19537916, 13913630, 16524926, 16462834, 12983611 13550185, 13810393, 14121009, 13065099, 11840910, 13903046, 15862017 13572659, 16294378, 13718279, 13657605, 17716305, 14480676, 13632717 14668670, 14063281, 14158012, 13736413, 13420224, 13812031, 12646784 16299830, 14512189, 10359307, 12755116, 17230530, 13616375, 14035825 13366199, 13427062, 18673342, 12861463, 13092220, 15862021, 17721717 13043012, 16619892, 13685544, 18325460, 13499128, 15862018, 19727057 13839336, 13866372, 13561750, 12718090, 13848402, 13725395, 5144934 12401111, 12796518, 13362079,

12917230, 12614359, 13042639, 14408859 13923374, 11732473, 14220725, 12621588, 13524899, 14480674, 14751895 13916709, 14781609, 14076523, 15905421, 12731940, 13343438, 17748835 14205448, 17082364, 14127231, 15853081, 14273397, 16844448, 14467061 12971775, 16864562, 14489591, 14497307, 12748538, 13872868, 10242202 14230270, 13931044, 13686047, 16382353, 14095982, 17333203, 19121548 13591624, 14523004, 13440516, 16794241, 13499412, 13035360, 14062795 12411746, 13040943, 13843646, 12905053, 18173592, 16794243, 13477790 14841409, 14609690, 14062797, 13059165, 12959852, 12345082, 16703112 13890080, 17333198, 16048375, 16450169, 12658411, 13780035, 14062793 19271438, 19259446, 13038684, 18740215, 16742095, 13742464, 13066936 14052474, 13060271, 13911821, 13457582, 7509451, 19710542, 13791364 12821418, 13502183, 13705338, 14237793, 16794239, 13554409, 15862024 13103913, 13645917, 12772404

Database Engine Version: 11.2.0.3.v3

What's New in Version 11.2.0.3.v3

This version adds support for the following:

- Oracle Database Patch Set Update (PSU) 11.2.0.3.14 (patch 20299017, released in April 2015)
- Installs additional Oracle Text knowledge bases from Oracle Database. Examples media (English and French)
- Provides access to DBMS_REPAIR through RDSADMIN.RDSADMIN_DBMS_REPAIR
- Grants ALTER DATABASE LINK, ALTER PUBLIC DATABASE LINK, EXEMPT ACCESS POLICY, EXEMPT IDENTITY POLICY, and EXEMPT REDACTION POLICY to master user

Baseline: Oracle Database Patch Set Update (PSU) 11.2.0.3.14 (April2015)

Bugs fixed: 13593999, 10350832, 19433746, 14138130, 12919564, 14198511, 13561951 13588248, 13080778, 20134036, 13804294, 16710324, 18031683, 12873183 16992075, 14193240, 14472647, 12880299, 13369579, 14799269, 13840704 14409183, 13492735, 14263036, 12857027, 13496884, 14263073, 16038929 13834436, 13015379, 17748833, 13732226, 16563678, 13866822, 20134034 13742434, 13944971, 12950644, 17748831, 12899768, 16929165, 16272008 13063120, 14613900, 13958038, 13503204, 13972394, 11877623, 17088068 13072654, 12395918, 16710753, 13429702, 13814739, 17343514, 13649031 13981051, 10256843, 15981698, 13901201, 12797765, 17333200, 19211724 12923168, 16761566, 13384182, 16279401, 13466801, 15996344, 14207163 13596581, 18673304, 13724193, 11063191, 13642044, 12940637, 19915271 12595606, 18641419, 14052871, 9163477, 15931756, 18262334, 13945708 12797420, 14123213, 13041324, 12865902, 15869211, 14003090, 16314468 16019955, 11708510, 17865671, 13026410, 14637368, 13737746, 13742438 15841373, 16347904, 16088176, 15910002, 19517437, 19827973, 16362358 16505333, 14398795, 14182835, 13579992, 11883252, 16344871, 10182005 10400244, 13742436, 14275605, 19197175, 9858539, 20477071, 14841812 16338983, 9703627, 13483354, 14393728, 14207317, 17165204, 20477069 12764337, 16902043, 14459552, 14191508, 14588746, 12964067, 19358317 20477440, 12780983, 12583611, 14383007, 14546575, 13476583, 15862016 13489024, 12985237, 17748830, 19554106, 14088346, 13448206, 19458377 16314466, 13419660, 18139695, 12591399, 14110275, 13430938, 13467683 17767676, 14548763, 19638161, 13424216, 12834027, 13632809, 13853126 13377816, 13036331, 14727310, 9812682, 12320556, 16747736, 13584130 16175381, 17468141, 12829021, 14138823, 15862019, 12794305, 14546673 12791981, 13503598, 13787482, 10133521, 12744759, 13399435, 18641461 19433747, 14023636, 13553883, 14762511, 9095696, 14343501, 12977562 13860201, 13257247, 14176879, 13783957, 16014985, 14480675, 12312133 13559697, 13146182, 16306019, 12974860, 9706792, 12940620, 13098318 13773133, 15883525, 16794244, 13340388, 13528551, 13366202, 12894807 13259364, 12747437, 13454210, 12748240, 13385346, 15987992, 13923995 16101465, 14571027, 13582702, 12784406, 13907462, 19769496, 13493847 13035804, 13857111, 13544396, 16710363, 10110625, 20134033, 14128555 12813641, 8547978, 14226599, 17478415, 17050888, 16923127, 17333197 9397635, 14007968, 13912931, 12693626, 12925089, 14189694, 17761775 12815057, 16721594, 13332439, 20477068, 19972198, 14038787,

11071989 14207902, 12596444, 14062796, 12913474, 20299010, 14390252, 13840711 13370330, 16314470, 14062794, 13358781, 12960925, 17333202, 9659614 13699124, 14546638, 13936424, 9797851, 19433745, 16794240, 14301592 13338048, 12938841, 12620823, 12656535, 12678920, 13719292, 14488943 14062792, 16850197, 14791477, 13807411, 16794238, 13250244, 12594032 15862022, 14098509, 15826962, 12612118, 9761357, 18096714, 19854461 14053457, 18436647, 13918644, 13527323, 10625145, 18173595, 12797620 19289642, 15862020, 13910420, 12780098, 13696216, 14774091, 14841558 10263668, 13849733, 16794242, 16944698, 15862023, 16056266, 13834065 20134035, 13853654, 14351566, 13723052, 18173593, 14063280, 13011409 13566938, 13737888, 13624984, 16024441, 17333199, 13914613, 17540582 14258925, 14222403, 14755945, 13645875, 12571991, 13839641, 14664355 12998795, 14469008, 13719081, 13361350, 14188650, 17019974, 13742433 14508968, 16314469, 16368108, 12905058, 6690853, 13647945, 16212405 12849688, 18641451, 13742435, 13464002, 18681866, 12879027, 13534412 18522512, 12585543, 12747740, 12535346, 13878246, 13790109, 16382448 12588744, 13916549, 13786142, 12847466, 13855490, 13551402, 12582664 19972199, 13871316, 14262913, 14657740, 17332800, 14558880, 14695377 13612575, 12912137, 13484963, 12387467, 14163397, 17437634, 13772618 19006849, 16694777, 13070939, 15994107, 14369664, 12391034, 13605839 12588237, 16279211, 16314467, 12945879, 15901852, 17762296, 14692762 12976376, 7276499, 12755231, 13680405, 13742437, 14589750, 14318397 11868640, 14644185, 13326736, 19309466, 13596521, 13001379, 12898558 13099577, 17752121, 13911711, 9873405, 18673325, 16372203, 16344758 11715084, 9547706, 16231699, 14040433, 12662040, 12617123, 14406648 17748832, 16530565, 12845115, 16844086, 13354082, 17748834, 13794550 13397104, 19537916, 13913630, 16524926, 16462834, 12983611, 13550185 13810393, 14121009, 13065099, 11840910, 13903046, 15862017, 13572659 16294378, 13718279, 13657605, 17716305, 14480676, 13632717, 14668670 14063281, 14158012, 13736413, 13420224, 13812031, 12646784, 16299830 18440047, 14512189, 10359307, 12755116, 14035825, 17230530, 13616375 13366199, 13427062, 18673342, 12861463, 15862021, 13092220, 17721717 13043012, 16619892, 13685544, 18325460, 13499128, 15862018, 19727057 13839336, 13866372, 13561750, 12718090, 13848402, 13725395, 12401111 5144934, 12796518, 13362079, 12917230, 12614359, 13042639, 14408859 13923374, 11732473, 14220725, 12621588, 13524899, 14480674, 14751895 13916709, 14781609, 14076523, 15905421, 12731940, 13343438, 14205448 17748835, 15853081, 17082364, 14127231, 14273397, 16844448, 14467061 12971775, 16864562, 20074391, 14489591, 14497307, 13872868, 12748538 10242202, 14230270, 13931044, 13686047, 16382353, 14095982, 17333203 19121548, 13591624, 14523004, 13440516, 16794241, 13499412, 13035360 14062795, 12411746, 13040943, 12905053, 13843646, 20296213, 18173592 16794243, 13477790, 14841409, 14609690, 14062797, 13059165, 12959852 12345082, 16703112, 13890080, 17333198, 16048375, 16450169, 12658411 13780035, 14062793, 19271438, 19259446, 13038684, 18740215, 16742095 13742464, 14052474, 13066936, 13060271, 13911821, 13457582, 7509451 19710542, 13791364, 12821418, 13502183, 13705338, 15856660, 14237793 16794239, 13554409, 15862024, 13103913, 13645917, 12772404

Database Engine Version: 11.2.0.4.v1

What's New in Version 11.2.0.4.v1

This version adds support for the following:

- Creating New Directories in the Main Data Storage Space (for version 11.2.0.4.v1 and later) (p. 251)
- Oracle PSU 11.2.0.4.1

Baseline: Oracle Database Patch Set Update (PSU) 11.2.0.4.1 (January2014)

Bugs fixed: 17432124, 16850630, 17551709, 13944971, 17811447, 13866822, 17811429, 16069901 16721594, 17443671, 17478514, 17612828, 17610798, 17239687, 17501491 17446237, 16450169, 17811438, 17288409, 17811456, 12905058, 17088068 16285691, 17332800

Database Engine Version: 11.2.0.4.v2 (Deprecated)

What's New in Version 11.2.0.4.v2

This version adds support for the following:

- Oracle Database Patch Set Update (PSU) 11.2.0.4.3 (patch 18522509, released July 2014)
- User access to DBMS_TRANSACTION package to clean-up failed distributed transactions
- Latest DST file (DSTv22 – patch 18759211, released in June 2014). This patch is incorporated by default only in new Oracle DB instances.
- Grants DBMS_REPUTIL to DBA role (upgrade to 11.2.0.4 revokes it from public)
- Privileges granted on DBMS_TRANSACTION, `v$pending_xatrans$`, and `v$xatrans$`
- Resolves a problem with DDL commands when user objects have “SYSTEM” in their names
- Installs schema objects to support XA Transactions, allowing transactions to be managed by an external transaction manager
- Permits truncation of temporary SYS and SYSTEM objects, allowing tools like LogMiner to function correctly

Baseline: Oracle Database Patch Set Update (PSU) 11.2.0.4.3 (July2014)

Bugs fixed: 17432124, 18759211, 18522509, 18031668, 17478514, 17752995, 17288409, 16392068, 17205719, 17811429, 17767676, 17614227 17040764, 17381384, 17754782, 17726838, 13364795, 17311728, 17389192 17006570, 17612828, 17284817, 17441661, 13853126, 17721717, 13645875 18203837, 17390431, 16542886, 16992075, 16043574, 17446237, 16863422 14565184, 17071721, 17610798, 17468141, 17786518, 17375354, 17397545 18203838, 16956380, 17478145, 16360112, 17235750, 17394950, 13866822 17478514, 17027426, 12905058, 14338435, 16268425, 13944971, 18247991 14458214, 16929165, 17265217, 13498382, 17786278, 17227277, 17546973 14054676, 17088068, 16314254, 17016369, 14602788, 17443671, 16228604 16837842, 17332800, 17393683, 13951456, 16315398, 18744139, 17186905 16850630, 17437634, 19049453, 17883081, 15861775, 17296856, 18277454 16399083, 16855292, 18018515, 10136473, 16472716, 17050888, 17865671 17325413, 14010183, 18554871, 17080436, 16613964, 17761775, 16721594 17588480, 17551709, 17344412, 18681862, 15979965, 13609098, 18139690 17501491, 17239687, 17752121, 17602269, 18203835, 17297939, 17313525 16731148, 17811456, 14133975, 17600719, 17385178, 17571306, 16450169 17655634, 18094246, 17892268, 17165204, 17011832, 17648596, 16785708 17477958, 16180763, 16220077, 17465741, 17174582, 18522509, 16069901 16285691, 17323222, 18180390, 17393915, 16875449, 18096714, 17238511

Database Engine Version: 11.2.0.4.v3

What's New in Version 11.2.0.4.v3

This version adds support for the following:

- Oracle Database Patch Set Update (PSU)11.2.0.4.4 (patch 19121551, released in October 2014)
- Latest DST file (DSTv23 – patch 19396455, released in Oct 2014). This patch is incorporated by default in new instances only.

Baseline: Oracle Database Patch Set Update (PSU) 11.2.0.4.4 (October2014)

Bugs fixed: 19396455, 18759211, 17432124, 16799735, 17288409, 17205719, 17811429, 17754782, 17726838, 13364795, 17311728 17284817, 17441661, 13645875, 18199537, 16992075, 16542886, 17446237 14565184, 17071721, 17610798, 17375354, 17449815, 17397545, 19463897 18230522, 17235750, 16360112, 13866822, 17982555, 17478514, 12905058 14338435, 13944971, 16929165, 12747740, 17546973, 14054676, 17088068 18264060, 17343514, 17016369, 17042658, 14602788, 14657740, 17332800 19211724, 13951456, 16315398, 17186905, 18744139, 16850630, 17437634 19049453, 18673304, 17883081, 18641419, 17296856, 18262334, 17006183 18277454, 17232014, 16855292, 10136473, 17705023, 17865671, 18554871 19121551, 17588480, 17551709, 17344412, 17842825, 18681862, 17390160 13955826, 13609098, 18139690, 17501491, 17239687, 17752121, 17299889 17602269, 18673325, 17313525, 17242746, 19544839, 17600719, 18191164 17571306, 19466309, 17951233, 18094246, 17165204, 17011832, 17040527 16785708, 16180763, 17477958, 17174582, 17465741, 18522509, 17323222 19463893, 16875449, 16524926, 17237521, 17596908, 17811438, 17811447 18031668, 16912439, 16494615, 18061914, 17545847, 17082359, 19554106 17614134, 17341326, 17891946, 19458377, 17716305, 17752995, 16392068 19271443, 17767676, 17614227, 17040764, 17381384, 18973907, 18673342 14084247, 17389192, 17006570, 17612828, 17721717, 13853126, 18203837 17390431, 17570240, 14245531, 16043574, 16863422, 19727057, 17468141 17786518, 17037130, 17267114, 18203838, 16198143, 16956380, 17478145 14829250, 17394950, 17027426, 16268425, 18247991, 19584068, 14458214 18436307, 17265217, 13498382, 16692232, 17786278, 17227277, 16042673 16314254, 17443671, 16228604, 16837842, 17393683, 17787259, 18009564 15861775, 16399083, 18018515, 16472716, 17050888, 14010183, 17325413 16613964, 17080436, 17036973, 17761775, 16721594, 18280813, 15979965 18203835, 17297939, 16731148, 17811456, 14133975, 17385178, 17586955 16450169, 17655634, 9756271, 17892268, 17648596, 16220077, 16069901 11733603, 16285691, 17587063, 18180390, 17393915, 18096714, 17238511 17824637, 14285317, 19289642, 14764829, 18328509, 17622427, 16943711 17346671, 18996843, 14852021, 17783588, 16618694, 17672719, 17546761

Database Engine Version: 11.2.0.4.v4

What's New in Version 11.2.0.4.v4

This version adds support for the following:

- Oracle Database Patch Set Update : 11.2.0.4.6 (patch 20299013, released April 2015)
- Installs additional Oracle Text knowledge bases from Oracle Database. Examples media (English and French)
- Provides access to DBMS_REPAIR through RDSADMIN.RDSADMIN_DBMS_REPAIR
- Grants ALTER DATABASE LINK, ALTER PUBLIC DATABASE LINK, EXEMPT ACCESS POLICY, EXEMPT IDENTITY POLICY, and EXEMPT REDACTION POLICY to master user

Baseline: Oracle Database Patch Set Update (PSU) 11.2.0.4.6 (April2015)

Bugs fixed: 17288409, 17798953, 18273830, 18607546, 17811429, 17205719, 20506699 17816865, 19972566, 17922254, 17754782, 16384983, 17726838, 13364795 16934803, 17311728, 17284817, 17441661, 17360606, 13645875, 18199537 16992075, 16542886, 17446237, 14015842, 17889549, 14565184, 19972569 17071721, 20299015, 17610798, 17375354, 17449815, 17397545, 19463897 18230522, 13866822, 17235750, 17982555, 16360112, 18317531, 17478514 19769489, 12905058, 14338435, 18235390, 13944971, 18641451, 20142975 17811789, 16929165, 18704244, 12747740, 18430495, 20506706, 17546973 14054676, 17088068, 17346091, 18264060, 17016369, 17042658, 17343514 14602788, 19972568, 19680952, 18471685, 19788842, 18508861, 14657740 17332800,

19211724, 13837378, 13951456, 16315398, 17186905, 18744139 19972564, 16850630, 18315328, 17437634, 19049453, 18673304, 17883081 19006849, 19915271, 19013183, 18641419, 17296856, 18674024, 18262334 17006183, 18277454, 16833527, 17232014, 16855292, 10136473, 17762296 14692762, 17705023, 18051556, 17865671, 17852463, 18554871, 17853498 19121551, 18334586, 19854503, 17551709, 19309466, 17588480, 19827973 17344412, 17842825, 18828868, 18681862, 18554763, 17390160, 18456514 16306373, 17025461, 13955826, 18139690, 11883252, 13609098, 17501491 17239687, 17752121, 17299889, 17602269, 19197175, 17889583, 18316692 17313525, 18673325, 12611721, 19544839, 18293054, 17242746, 18964939 17600719, 18191164, 19393542, 17571306, 18482502, 19466309, 17951233 17649265, 18094246, 19615136, 17040527, 17011832, 17165204, 18098207 16785708, 16870214, 17465741, 16180763, 17174582, 17477958, 12982566 16777840, 18522509, 20631274, 16091637, 17323222, 19463893, 16595641 16875449, 12816846, 16524926, 17237521, 18228645, 18282562, 17596908 19358317, 17811438, 17811447, 17945983, 18762750, 17156148, 18031668 16912439, 17184721, 16494615, 18061914, 17282229, 17545847, 18331850 18202441, 17082359, 18723434, 19554106, 17614134, 13558557, 17341326 14034426, 17891946, 18339044, 17716305, 19458377, 17752995, 16392068 19271443, 17891943, 18092127, 17258090, 17767676, 16668584, 18384391 17614227, 17040764, 16903536, 17381384, 14106803, 15913355, 18973907 18356166, 18673342, 17389192, 14084247, 16194160, 17612828, 17006570 20506715, 17721717, 13853126, 17390431, 18203837, 17570240, 14245531 16043574, 16863422, 17848897, 17877323, 18325460, 19727057, 17468141 17786518, 17912217, 16422541, 19972570, 17267114, 17037130, 18244962 18765602, 18203838, 18155762, 16956380, 16198143, 17246576, 17478145 17394950, 14829250, 18189036, 18641461, 18619917, 17835627, 17027426 16268425, 18247991, 19584068, 14458214, 18436307, 17265217, 17634921 13498382, 16692232, 17786278, 17227277, 16042673, 16314254, 17443671 18000422, 16228604, 16837842, 17571039, 17393683, 16344544, 17787259 18009564, 20074391, 14354737, 15861775, 18135678, 18614015, 16399083 18362222, 18018515, 16472716, 17835048, 17050888, 17936109, 14010183 17325413, 18747196, 17080436, 16613964, 17036973, 17761775, 16579084 16721594, 17082983, 18384537, 18280813, 20296213, 17302277, 16901385 18084625, 15979965, 15990359, 18203835, 17297939, 17811456, 16731148 13829543, 14133975, 17215560, 17694209, 18091059, 17385178, 8322815 17586955, 18441944, 17201159, 16450169, 9756271, 17655634, 19730508 17892268, 18868646, 17648596, 16220077, 16069901, 11733603, 16285691 17587063, 18180390, 16538760, 18193833, 17348614, 17393915, 17957017 17274537, 18096714, 17308789, 17238511, 18436647, 17824637, 14285317 19289642, 14764829, 17622427, 18328509, 16571443, 16943711, 14368995 18306996, 17346671, 14852021, 18996843, 17783588, 16618694, 17853456 18674047, 17672719, 18856999, 12364061, 18783224, 17851160, 17546761

Database Engine Version: 12.1.0.1.v1

What's New in Version 12.1.0.1.v1

This version adds support for:

- Database Patch Set Update : 12.1.0.1.6 (Jan 2015) (19769486)

Baseline: Oracle Database Patch Set Update : 12.1.0.1.6 (Jan 2015) (19769486)

Bugs fixed: 18093615, 17716305, 17257820, 17034172, 16694728, 16042673, 18096714 17439871, 16320173, 14664684, 17762256, 18002100, 18436307, 16450169 17006570, 17753428, 17552800, 15994107, 17441661, 17305959, 18362376 17997255, 17710315, 14506328, 17806676, 17443596, 17040764, 16849982 16837842, 14010183, 18393024, 16845022, 16228604, 17446564, 17042658 14536110, 17579911, 18262334, 17311728, 17391312, 17244462, 16935643 18641419, 17039360, 14355775, 18155703, 16672859, 18229326, 17080436 17912217, 16788832, 16039096, 16570023, 18099539, 14123213, 17174391 17405549, 17830435, 17249820, 16946990, 16589507, 16924879, 16874123 17750832, 16784143, 15987992, 17346196, 16901482, 16859937, 17898041 17068536, 16910001, 17946998, 16527374, 17394724, 17572720, 16703112 17490498, 16433869, 16186165,

16170787, 16524968, 17032726, 16543323 17349104, 18355572, 17888553, 16575931, 18061914, 16070351, 17088068 16888264, 16448848, 16863422, 17443671, 18308576, 16911800, 16517900 16825779, 17019974, 16707927, 17551812, 14576755, 17263661, 17325413 17446849, 16465149, 17184677, 16689109, 16705020, 18889652, 17828499 16964279, 15953721, 17205719, 18603606, 18121501, 16757934, 16864562 16782193, 17436936, 15996344, 17037526, 17260090, 19556045, 17216406 17659488, 16485876, 16709437, 17898730, 18641461, 17174582, 16796277 17421502, 17534365, 16921340, 16784167, 18292893, 16660558, 16793174 16371304, 20074391, 17570606, 16943711, 16674666, 19197175, 16697600 17848854, 18522516, 17797837, 17716565, 16456371, 16347068, 16181570 19121550, 17516005, 16275522, 16475788, 16683859, 17491753, 16427054 16227068, 17753514, 16479182, 18554871, 17051636, 16263492, 16551086 18856947, 19866250, 16406802, 16433745, 16681689, 17614134, 17364702 17171530, 17298973, 16212405, 19049453, 18189497, 16443657, 16855202 18078926, 18244962, 17462687, 16087650, 16313881, 16992075, 17082983 17359546, 14595800, 16715647, 19554106, 17362796, 17777061, 16392068 17761775, 16977973, 17158214, 14197853, 16712618, 12911115, 17922172 16524071, 16856570, 17050888, 16410570, 17210416, 13866822, 18513099 16372203, 17867137, 16101465, 15914210, 16459685, 16802693, 16195633 16978185, 19309466, 17983206, 16787973, 16850996, 16178562, 16838328 16503473, 18126350, 13782826, 18439152, 17537657, 17721717, 17489214 16362358, 16994576, 17600719, 17461374, 16969016, 17571945, 16444683 16928832, 16929165, 16710753, 16864359, 16679874, 18031528, 16585173 15986012, 17467075, 17735933, 14852021, 16191248, 19692901, 16173738 17797453, 17343514, 16495802, 17324822, 16619249, 19297295, 16590848 15921906, 16986421, 17316776, 16576250, 16730813, 16433540, 16663303 18420490, 16619979, 17897716, 17016479, 16457621, 16675739, 17341326 17981677, 17005047, 17442449, 16795944, 16668226, 16698577, 16621274 17330580, 18348157, 17393683, 17817656, 16634384, 16465158, 16816103 16910734, 16584684, 16936008, 16347904, 16512817, 17273253, 16902138 17179261, 17810688, 16864048, 17468141, 17226980, 17883081, 16682595 16473934, 16762985, 16864864, 16721594, 16946613, 16972213, 16855292 17026503, 16964686, 17860549, 16674842, 13771513, 16061921, 17235750 16842274, 16913149, 16769019, 17000176, 15931910, 17572525, 17478145 14237793, 19248799, 17976046, 17289787, 16919176, 16613964, 17217040 16462834, 18092561, 16617325, 17308691, 16733884, 16483559, 16057129 16286774, 16822629, 17596344, 19289642, 17954431, 18423374, 16993424 17605522, 17280117, 19769486, 18436647, 8352043, 18973907, 16772060 16790307, 16991789, 17608025, 19006849, 18082092, 20128874, 16603924 18148383, 17182200, 16784901, 16912439, 18641451, 13521413, 17767676 17478811, 16836849, 16007562, 16851772, 16663465, 17786278, 17027533 16675710, 17437634, 19458377, 17610418, 17465741, 15905421, 17892268 16523150, 16741246, 16930325, 17982838, 17390431, 17974104

Database Engine Version: 12.1.0.1.v2

What's New in Version 12.1.0.1.v2

This version adds support for the following:

- Oracle Database Patch Set Update: 12.1.0.1.7 (patch 20299016, released April 2015)
- Installs additional Oracle Text knowledge bases from Oracle Database. Examples media (English and French)
- Provides access to DBMS_REPAIR through RDSADMIN.RDSADMIN_DBMS_REPAIR
- Grants ALTER DATABASE LINK, ALTER PUBLIC DATABASE LINK, EXEMPT ACCESS POLICY, EXEMPT IDENTITY POLICY, and EXEMPT REDACTION POLICY to master user

Baseline: Oracle Database Patch Set Update: 12.1.0.1.7 (Patch 20299016, released April 2015)

Bugs fixed: 16406802, 16576250, 17605522, 16433745, 18439152, 16465158, 16523150 18002100, 17830435, 16101465, 17316776, 16660558, 17797837, 16433540 16462834, 16924879, 16842274, 16663465, 16465149, 18603606, 14010183 16705020, 17912217, 14664684, 15921906, 17817656,

17040764, 17478145 17039360, 19358317, 16991789, 17777061, 18148383, 16485876, 16787973 17405549, 16433869, 18641419, 16715647, 17467075, 13771513, 16911800 16456371, 18420490, 16712618, 17216406, 16825779, 19297295, 16921340 17797453, 16443657, 16994576, 16479182, 17441661, 17465741, 16993424 16788832, 16888264, 17437634, 14237793, 16910001, 16838328, 16689109 17298973, 18126350, 16795944, 19554106, 19458377, 16512817, 16762985 17596344, 17516005, 16524968, 16757934, 18641461, 16946613, 16978185 16495802, 16543323, 19309466, 17171530, 16935643, 17308691, 17571945 16782193, 17226980, 18641451, 17982838, 16855202, 16683859, 16619979 17888553, 16173738, 17364702, 16837842, 17005047, 16772060, 16710753 14197853, 17205719, 17848854, 16459685, 17860549, 17976046, 17750832 16347904, 17359546, 17080436, 17898041, 18155703, 18078926, 16007562 16410570, 17551812, 17390431, 16856570, 16613964, 16802693, 19556045 16061921, 18355572, 16551086, 17954431, 16668226, 17217040, 17600719 17184677, 16527374, 20074391, 16228604, 16042673, 14576755, 17393683 19006849, 17174582, 18393024, 17210416, 17000176, 18121501, 18348157 16697600, 17436936, 17034172, 17762296, 16450169, 17019974, 17974104 17898730, 18423374, 15994107, 15914210, 17289787, 16191248, 15986012 16679874, 16570023, 17442449, 15905421, 15996344, 19866250, 16475788 17489214, 17572525, 20296213, 14595800, 16263492, 15931910, 17608025 17659488, 17753428, 17391312, 17027533, 17311728, 17753514, 18096714 17897716, 16681689, 17478811, 16320173, 18554871, 16178562, 16864562 16362358, 16855292, 16039096, 17026503, 20299016, 18093615, 17324822 18513099, 16864359, 16698577, 17806676, 17867137, 20328279, 16850996 16901482, 17182200, 17892268, 17257820, 16603924, 16930325, 16212405 19248799, 16849982, 15987992, 16457621, 17443671, 16473934, 16913149 18308576, 17158214, 18856947, 18889652, 16707927, 16275522, 14355775 18099539, 16816103, 16912439, 16584684, 16517900, 16851772, 16859937 16919176, 17349104, 16741246, 17537657, 16972213, 17552800, 18522516 17341326, 17346196, 16928832, 16087650, 16585173, 19049453, 17735933 16796277, 16675739, 17273253, 13521413, 18061914, 17981677, 16822629 16372203, 16845022, 8352043, 17174391, 17810688, 16977973, 16634384 16672859, 16392068, 16863422, 18244962, 17037526, 17491753, 16682595 16575931, 17260090, 18973907, 17490498, 17088068, 17330580, 19915271 16195633, 16936008, 16181570, 17762256, 17721717, 16733884, 17534365 18436307, 17983206, 16793174, 18092561, 17922172, 19121550, 17716565 17446849, 16964686, 16617325, 17761775, 17461374, 16721594, 17179261 18262334, 17610418, 18292893, 16902138, 17042658, 16483559, 15953721 18229326, 16986421, 17305959, 16874123, 16769019, 17006570, 17032726 16070351, 19197175, 16371304, 16964279, 16864864, 17249820, 19692901 17446564, 17343514, 17325413, 17421502, 16170787, 17462687, 17362796 17439871, 16730813, 17883081, 17579911, 12911115, 18082092, 16347068 17946998, 16313881, 16227068, 17997255, 17394724, 17443596, 16836849 16969016, 14852021, 17716305, 17051636, 17244462, 19289642, 17767676 17786278, 18189497, 17572720, 17082983, 16444683, 16621274, 16524071 16709437, 14123213, 14506328, 16943711, 17710315, 13866822, 16503473 16590848, 16784143, 16057129, 18031528, 16619249, 16589507, 16674666 13782826, 19769486, 16790307, 16675710, 17828499, 16864048, 16694728 16910734, 17614134, 17263661, 16448848, 17050888, 16286774, 20128874 17280117, 18362376, 16427054, 16992075, 17016479, 16784167, 16663303 16674842, 17468141, 18436647, 16186165, 14536110, 16946990, 17570606 17235750, 16703112, 16784901, 17068536, 16929165

Database Engine Version: 12.1.0.2.v1

What's New in Version 12.1.0.2.v1

This version adds support for the following:

- Oracle Database Patch Set Update: 12.1.0.2.3 (patch 20299023, released April 2015)
- The In-Memory option allows storing a subset of data in an in-memory column format optimized for performance.
- Installs additional Oracle Text knowledge bases from Oracle Database. Examples media (English and French)
- Provides access to DBMS_REPAIR through RDSADMIN.RDSADMIN_DBMS_REPAIR
- Grants ALTER DATABASE LINK, ALTER PUBLIC DATABASE LINK, EXEMPT ACCESS POLICY, EXEMPT IDENTITY POLICY, and EXEMPT REDACTION POLICY to master user

Note
Version 12.1.0.2.v1 supports Enterprise Edition only.

Baseline: Oracle Database Patch Set Update : 12.1.0.2.3 (20299023)

Bugs fixed: 19189525, 19065556, 19075256, 19723336, 19077215, 19865345, 18845653 19280225, 19524384, 19248799, 18988834, 19048007, 18288842, 19238590 18921743, 18952989, 16870214, 19928926, 19134173, 19180770, 19018206 19197175, 19149990, 18849537, 19730508, 19183343, 19012119, 19001390 18202441, 19067244, 19189317, 19644859, 19358317, 19390567, 20074391 19279273, 19706965, 19068970, 19841800, 19512341, 14643995, 19619732 20348653, 18607546, 18940497, 19670108, 19649152, 19065677, 19547370 18948177, 19315691, 19637186, 19676905, 18964978, 19035573, 19176326 18967382, 19174430, 19176223, 19532017, 18674047, 19074147, 19054077 19536415, 19708632, 19289642, 20425790, 19335438, 18856999, 19371175 19468347, 19195895, 19154375, 16359751, 18990693, 19439759, 19769480 19272708, 19978542, 19329654, 19873610, 19174521, 19520602, 19382851 19658708, 19304354, 19052488, 19291380, 18681056, 19896336, 17835294 19076343, 19791377, 19068610, 19561643, 18618122, 20440930, 18456643 18909599, 19487147, 19143550, 19185876, 19016730, 18250893, 20347562 19627012, 16619249, 18354830, 19577410, 19687159, 19001359, 19174942 19518079, 18610915, 18674024, 18306996, 19309466, 19081128, 19915271 19157754, 19058490, 20284155, 18791688, 18885870, 19303936, 19434529 19018447, 18417036, 19597439, 20235511, 19022470, 18964939, 19430401 19044962, 19385656, 19501299, 17274537, 19409212, 19440586, 19606174 18436647, 19023822, 19684504, 19178851, 19124589, 19805359, 19024808 19597583, 19155797, 19393542, 19050649, 19028800

Microsoft SQL Server on Amazon RDS

Amazon RDS supports DB instances running several editions of Microsoft SQL Server 2008 R2 and SQL Server 2012. You can create DB instances and DB snapshots, point-in-time restores and automated or manual backups. DB instances running SQL Server can be used inside a VPC. You can also use SSL to connect to a DB instance running SQL Server, and you can use TDE to encrypt data at rest. Amazon RDS currently supports Multi-AZ deployments for SQL Server using SQL Server Mirroring as a high-availability, failover solution.

In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application such as Microsoft SQL Server Management Studio. Amazon RDS does not allow direct host access to a DB instance via Telnet, Secure Shell (SSH), or Windows Remote Desktop Connection. When you create a DB instance, you are assigned to the *db_owner* role for all databases on that instance, and you will have all database-level permissions except for those that are used for backups (Amazon RDS manages backups for you).

Before creating a DB instance, you should complete the steps in the Setting Up for Amazon RDS (p. 7) section of this guide.

Common Management Tasks for SQL Server on Amazon RDS

These are the common management tasks you perform with an Amazon RDS SQL Server DB instance, with links to information about each task:

- For planning information, such as SQL Server versions, storage engines, security, and features supported in Amazon RDS, see Planning Your SQL Server DB Instance on Amazon RDS (p. 302).
- If you are creating a DB instance for production purposes, you should understand how instance classes, storage, and Provisioned IOPS work in Amazon RDS. For more information about DB instance classes, see DB Instance Class (p. 65) For more information about Amazon RDS storage, see Amazon RDS Storage Types (p. 77). For more information about Provisioned IOPS, see Amazon RDS Provisioned IOPS Storage to Improve Performance (p. 82).

- A production DB instance should also use Multi-AZ deployments. All Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances. Multi-AZ deployments for SQL Server is implemented using SQL Server's native Mirroring technology. For more information about Multi-AZ deployments, see High Availability (Multi-AZ) (p. 71). For more information on SQL Server's Multi-AZ using Mirroring, see Planning your Multi-AZ Deployments Using SQL Server Mirroring (p. 307).
- There are prerequisites you must complete before you create your DB instance. For example, DB instances are created by default with a firewall that prevents access to it. You therefore must create a security group with the correct IP addresses and network configuration you will use to access the DB instance. The security group you need to create will depend on what EC2 platform your DB instance is on, and whether you will be accessing your DB instance from an EC2 instance. For more information about the two EC2 platforms supported by Amazon RDS, *EC2-VPC* and *EC2-Classic*, see Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 563). In general, if your DB instance is on the *EC2-Classic* platform, you will need to create a DB security group; if your DB instance is on the *EC2-VPC* platform, you will need to create a VPC security group. For more information about security groups, see Amazon RDS Security Groups (p. 110) or the Setting Up for Amazon RDS (p. 7) section of this guide.
- If your AWS account has a default VPC (a default virtual private network), then your DB instance will automatically be created inside the default VPC. If your account does not have a default VPC and you want the DB instance to be inside a VPC, you must create the VPC and subnet groups before you create the DB instance. For more information about determining if your account has a default VPC, see Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 563). For more information about using VPCs with Amazon RDS, see Using Amazon RDS with Amazon Virtual Private Cloud (VPC) (p. 562).
- If your DB instance is going to require specific database parameters or options, you should create the parameter or option groups before you create the DB instance. For more information on parameter groups, see Working with DB Parameter Groups (p. 523). For more information on options for SQL Server, see Appendix: Options for SQL Server Database Engine (p. 350).
- After creating a security group and associating it to a DB instance, you can connect to the DB instance using any standard SQL client application such as Microsoft SQL Server Management Studio. For more information on connecting to a DB instance, see Connecting to a DB Instance Running the SQL Server Database Engine (p. 321).
- You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots. For information, see Backing Up and Restoring (p. 495).
- You can monitor an instance through actions such as viewing the SQL Server logs, CloudWatch Amazon RDS metrics, and events. For information, see Monitoring Amazon RDS (p. 571).

There are also several appendices with useful information about working with Amazon RDS SQL Server DB instances:

- For information on common DBA tasks for SQL Server on Amazon RDS, see Appendix: Common DBA Tasks for SQL Server (p. 342).
- For information on the options that you can use with SQL Server on Amazon RDS, see Appendix: Options for SQL Server Database Engine (p. 350).

Planning Your SQL Server DB Instance on Amazon RDS

You can chose the version of SQL Server you want to have on your DB instance. Amazon RDS supports DB instances running several editions of Microsoft SQL Server 2008 R2 and SQL Server 2012. You should also be aware of the limits for SQL Server DB instances.

Topics

- General Limits for SQL Server DB Instances (p. 303)
- Support for SQL Server Features on Amazon RDS (p. 359)
- SQL Server Licensing (p. 306)
- Planning your Multi-AZ Deployments Using SQL Server Mirroring (p. 307)
- Database Engine Version Management (p. 309)
- Supported SQL Server Roles and Permissions (p. 310)
- Using SSL with a SQL Server DB Instance (p. 310)
- Using the TDE Option to Encrypt Data at Rest (p. 312)

General Limits for SQL Server DB Instances

The Amazon RDS implementation of SQL Server on a DB instance have some limitations you should be aware of:

- The maximum number of databases on a single Microsoft SQL Server DB Instance is 30.
- Databases cannot be renamed.
- The maximum storage size for a Microsoft SQL Server DB Instance is 4 TB for all instances except the SQL Server Express edition, which limits storage to a total of 300 GB.

  If you have a scenario that requires a larger amount of storage, it is possible to use sharding across multiple DB instances to get around this limit. This approach requires data-dependent routing logic in applications that connect to the sharded system, so that data gets queried from and written to the appropriate shard. You can either use an existing framework like Hibernate Shards or write custom code to enable this. If you do choose to use an existing framework, it must not require any components to be installed on the same server as the DB instance. For an example of a sharding solution using an existing framework, see Using an Example of Sharding with Hibernate.
- Because of the extensibility limitations of striped storage attached to Windows Server, Amazon RDS does not currently support increasing storage on a SQL Server DB Instance. We recommend that you provision storage according to anticipated future storage growth. If you need to increase the storage of a SQL Server DB Instance, you will need to export the data, create a new DB Instance with increased storage, and then import the data into the new DB Instance. For more information, go to the RDS SQL Server Data Migration Guide.
- The minimum storage size for a Microsoft SQL Server DB Instance is 20 GB for the Microsoft SQL Server Express and Web Editions and 200 GB for the Standard and Enterprise Editions.
- A newly created SQL Server DB instance does not contain a database. The instance has one master user account with the name and password you specified when you created the DB instance that you can use to create users and grant permissions. You must use a SQL Server tool such as SQL Server Management Studio to log in as the master user, and then use SQL Server commands and SQL statements to add the users and elements required by your applications to store and retrieve data in the DB instance
- To import SQL Server data into a DB instance, follow the information in the Importing and Exporting SQL Server Data (p. 332) section. You cannot use the BACKUP and RESTORE commands to import data into a DB instance because Amazon RDS does not allow OS-level access that would enable you to place files in a physical location that the database engine could access. You also cannot import data using the Copy Database Wizard in SQL Server Management Studio because the tool requires *sysadmin* privilege on the source and destination servers and this permission is not available to the master user account for a DB instance.
- Because of limitations in Microsoft SQL Server, restoring to a point in time before successful execution of a DROP DATABASE may not reflect the state of that database at that point in time. For example, the dropped database will typically be restored to its state up to 5 minutes before the DROP DATABASE command was issued, which means that you will not be able to restore the transactions made during those few minutes on your dropped database. To work around this, you can reissue the DROP

DATABASE command after the restore operation is completed. Note that dropping a database removes the transaction logs for that database.

- The *db.t1.micro* DB instance class has limited resources and is best used for testing. For example, the *db.t1.micro* DB instance class does not have enough resources for a full implementation of SQL Server 2012.
- While Amazon RDS doesn't support some features of SQL Server, you can run SQL Server components in an Amazon EC2 instances with EBS storage, pursuant to Microsoft licensing policies. This includes components such as SQL Server Analysis Services, SQL Server Integration Services, SQL Server Reporting Services, Data Quality Services, and Master Data Services.
- System time is maintained in UTC. We do not support changing the timezone for RDS SQL Server DB instances. Depending on your use case, you may be able to use the `datetimeoffset` data type to convert system time (UTC) to your local time zone. For more information on the `datetimeoffset` data type, see Date and Time Functions in the Microsoft SQL Server documentation.
- Some ports are reserved for Amazon RDS use and cannot be used when creating a DB instance.

Support for SQL Server Features on Amazon RDS

The following list shows a subset of the key database engine features that are currently supported by the 2008 R2 version of SQL Server. For a complete list of features supported by the 2008 R2 SQL Server database engine, go to Features Supported by the Editions of SQL Server.

- Core database engine features
- SQL Server development tools:
  - Visual Studio integration
  - IntelliSense
- SQL Server management tools:
  - SQL Server Management Studio (SMS)
  - sqlcmd
  - SQL Server Profiler (client side traces; workaround available for server side)
  - SQL Server Migration Assistant (SSMA)
  - Database Engine Tuning Advisor
  - SQL Server Agent
- Safe CLR
- Full-text search (except semantic search)
- SSL
- Transparent Data Encryption (Enterprise Edition only)
- Spatial and location features
- Change Tracking
- Database Mirroring
- The ability to use an Amazon RDS SQL DB instance as a data source for Reporting, Analysis, and Integration Services

In addition to the features above, the following list shows a subset of the key database engine features that are currently supported by the 2012 version of SQL Server. For more information on SQL Server 2012, see What's New in SQL Server 2012.

- Columnstore indexes (Enterprise Edition)
- Online Index Create, Rebuild and Drop for XML, varchar(max), nvarchar(max), and varbinary(max) data types (Enterprise Edition)
- Flexible Server Roles

- Partially Contained Databases
- Sequences
- Transparent Data Encryption (Enterprise Edition only)
- THROW statement
- New and enhanced spatial types
- UTF-16 Support
- ALTER ANY SERVER ROLE server-level permission

Amazon RDS currently does not support the following SQL Server features:

- Maintenance Plans
- Database Mail
- Distributed Queries (i.e., Linked Servers)
- Service Broker
- Database Log Shipping
- Windows Authentication
- Change Data Capture (CDC) - Consider using Change Tracking as an alternative to CDC.
- Replication
- The ability to run Reporting, Analysis, Integration, or Master Data Services on the same server as the DB instance. If you need to do this, we recommend that you either install SQL Server on an EC2 instance or use an on-premise SQL Server instance to act as the Reporting, Analysis, Integration, or Master Data Services server.
- Performance Data Collector
- Additional T-SQL endpoints
- Distribution Transaction Coordinator (MSDTC)
- WCF Data Services
- FILESTREAM support
- Policy-Based Management
- SQL Server Audit
- BULK INSERT and OPENROWSET(BULK...) features. These must be run from client-based server storage.
- Data Quality Services
- Instant file initialization
- Always On (2012 Enterprise Edition)
- File tables
- Server level triggers

Some SQL Server parameters have changed in SQL Server 2012.

- The following parameters have been removed from SQL Server 2012: `awe enabled`, `precompute rank`, and `sql mail xps`. These parameters were not modifiable in SQL Server DB Instances and their removal should have no impact on your SQL Server use.
- A new `contained database authentication` parameter in SQL Server 2012 supports "partially contained databases." When you enable this parameter and then create a partially contained database, an authorized user's user name and password is stored within the partially contained database instead of in the master database. For more information about partially contained databases, go to Contained Databases.

SQL Server Licensing

Currently, Amazon RDS offers two licensing options for SQL Server, License Included and License Mobility (Bring Your Own License). This section explains each.

In accordance with Microsoft's usage rights, SQL Server Web Edition can be used only to support public and Internet-accessible web pages, websites, web applications, and Web services. For more information, go to AWS Service Terms.

Note
The secondary instance of a SQL Server Multi-AZ deployment (mirroring) is passive and does not take writes or provide reads until a failover occurs. Therefore, you do not need a license for this secondary instance. For more information on SQL Server 2012 licensing, see the SQL Server 2012 Licensing Reference Guide from Microsoft.

License Included

Amazon RDS uses the *License Included* service model for DB instances running the Microsoft SQL Server Express Edition, Microsoft SQL Server Web Edition, Microsoft SQL Server Standard Edition (SE), and Microsoft SQL Server Enterprise Edition (EE). In the License Included service model, you do not need to separately purchase Microsoft SQL Server licenses. License Included pricing is inclusive of software license, underlying hardware resources, and Amazon RDS management capabilities. In this model, the license is held by AWS and is included in the price of the DB instance.

Note
The License Included option for Microsoft SQL Server Enterprise Edition is only available in the US West (Oregon), US East (N. Virginia), and EU (Ireland) regions and is available on R3.2xlarge, R3.4xlarge, and R3.8xlarge instance classes.

License Mobility (Bring Your Own License)

Microsoft's License Mobility program allows Microsoft customers to easily move current on-premises Microsoft Server application workloads to Amazon Web Services (AWS), without any additional Microsoft software license fees. This benefit is available to Microsoft Volume Licensing (VL) customers with eligible server applications covered by active Microsoft Software Assurance (SA) contracts. Currently, Microsoft SQL Server Standard Edition and Microsoft SQL Server Enterprise Edition are the eligible Database editions for this program. Refer to Microsoft's Product Use Rights for the latest licensing terms.

Licensing for SQL Server 2012

The following table shows the license models that are supported for each SQL Server 2012 version.

SQL Server 2012 Engine Type	license-included	bring-your-own-license
Enterprise Edition	Yes*	Yes
Standard Edition	Yes	Yes
Web Edition	Yes	No
Express Edition	Yes	No

Note
* The License Included option for Microsoft SQL Server Enterprise Edition is only available in the US West (Oregon), US East (N. Virginia), and EU (Ireland) regions and is available on R3.2xlarge, R3.4xlarge, and R3.8xlarge instance classes.

For more information on supported licensing methods on Amazon RDS for SQL Server, see SQL Server License Requirements for Microsoft License Mobility.

Planning your Multi-AZ Deployments Using SQL Server Mirroring

Amazon RDS supports SQL Server Multi-AZ deployments using SQL Server Mirroring. All Multi-AZ implementations provide increased availability, data durability, and fault tolerance for DB instances. In the event of planned database maintenance or unplanned service disruption, Amazon RDS will automatically failover to the up-to-date standby such that database operations can resume quickly without manual intervention. The primary and standby instances use the same endpoint whose physical network address transitions to the mirror as part of the failover process, so you do not have to reconfigure your application or set up multiple endpoints when a failover occurs. For more information about Multi-AZ, see High Availability (Multi-AZ) (p. 71).

> **Note**
> SQL Server Multi-AZ using Mirroring is currently available in the US East (N. Virginia), US West (Oregon), and EU (Ireland) AWS regions. We plan to support other regions in the future.

Multi-AZ deployments are available for SQL Server Standard and Enterprise Edition with SQL Server 2008R2 and SQL Server 2012. Multi-AZ with Mirroring supports one standby mirror. You can enable Multi-AZ using the RDS console or by setting the Multi-AZ Deployment for SQL Server Using the Mirroring Option (p. 353) in an option group and then associating that option group with your DB instance. For more information on working with Mirroring, see Working with SQL Server Multi-AZ with Mirroring (p. 331)

With a Multi-AZ deployment using Mirroring, Amazon RDS manages failover by actively monitoring your Multi-AZ deployment and proactively initiating a failover when a problem with your primary occurs. Failover does not occur unless the standby and primary are fully in sync. In addition, Amazon RDS will actively reestablish your Multi-AZ deployment by automatically repairing unhealthy DB instances and reestablishing synchronous replication. There is nothing for you to manage; Amazon RDS handles the primary, the Mirroring witness, and the standby instance for you. When you set up SQL Server Multi-AZ, all databases on the instance will be mirrored automatically. The Amazon RDS console, CLI, and API can show what Availability Zone the standby instance is located in.

Multi-AZ deployments, including Multi-AZ with Mirroring maintain all databases on the same node. If a database on the primary host fails over, all your SQL Server databases will failover as one atomic unit to your standby host. This allows Amazon RDS to provision a new, healthy host and replace the unhealthy host.

SQL Server Multi-AZ Deployment Recommendations

- For databases used in production or pre-production we recommend Multi-AZ deployments for high availability, Provisioned IOPS for fast, consistent performance, and instance classes (m1.large and larger) that are optimized for Provisioned IOPS.
- Users, Logins, and Permissions are automatically replicated for you on the standby mirror . You don't need to worry about recreating them. User-defined server roles (a SQL 2012 feature) are not replicated in Multi-AZ instances.
- To use SQL Server Mirroring with a SQL Server DB instance in a VPC, you must create a DB subnet group that has 3 subnets in distinct Availability Zones. You must then assign the DB subnet group to the SQL Server DB instance that is being mirrored.
- You cannot select the Availability Zone for the standby instance, so you should deploy any applications using the SQL Server databases into all AZs of the region to take advantage of this feature.
- Note that you cannot configure the standby to accept database read activity.
- Failover times will be affected by the time it takes to complete the recovery process. Large transactions will increase the failover time.

- For best performance, do not enable mirroring during a large data load operation. If you want your data load to be as fast as possible, complete the loading before you convert your DB instance to a Multi-AZ deployment.
- Your application that accesses the SQL Server databases should have exception handling that will catch connection errors. The following code sample shows a try/catch block that will catch a communication error.

```
for (int iRetryCount = 0; (iRetryCount < RetryMaxAttempts && keepInserting);
 iRetryCount++)
{
 using (SqlConnection connection = new SqlConnection(DatabaseConnString))
 {
  using (SqlCommand command = connection.CreateCommand())
  {
   command.CommandText = "INSERT INTO SOME_TABLE VALUES ('SomeValue');";

   try
   {
    connection.Open();

    while (keepInserting)
    {
     command.ExecuteNonQuery();
     intervalCount++;
    }
    connection.Close();
   }
   catch (Exception ex)
   {
    Logger(ex.Message);
   }
  }
 }

 if (iRetryCount < RetryMaxAttempts && keepInserting)
 {
  Thread.Sleep(RetryIntervalPeriodInSeconds * 1000);
 }

}
```

- If you created SQLAgent jobs, these will need to be recreated in the secondary, as these jobs are stored in the msdb and this database cannot be replicated via Mirroring. You should create the jobs first in the original primary, then fail over, and create the same jobs in the new primary.
- You should not use the `Set Partner Off` command when working with Multi-AZ instances. For example, DO NOT do the following:

```
alter database db1 set partner off
go
```

- You should not set the recovery mode to `simple`. For example, DO NOT do the following:

```
alter database db1 set recovery simple
go
```

- You should not use the `DEFAULT_DATABASE` parameter when creating new logins on Multi-AZ DB instances as these settings cannot be applied to the standby mirror. For example, DO NOT do the following:

```
CREATE LOGIN [test_dba] WITH PASSWORD=foo, DEFAULT_DATABASE=[db2]
GO
or
ALTER LOGIN [test_dba] SET DEFAULT_DATABASE=[db3]
GO
```

- Cross-region Multi-AZ is not currently supported. If you are interested in cross-region disaster recovery, you are encouraged to try our cross-region snapshot copy feature that is available today.
- You may observe elevated latencies relative to a standard DB Instance deployment in a single Availability Zone as a result of the synchronous data replication performed on your behalf.

Video Introduction to SQL Server Multi-AZ Deployments

The video "Getting Started with Multi-AZ Deployments in Amazon RDS for SQL Server" is available here.

Database Engine Version Management

With Amazon RDS, you can control when to upgrade your SQL Server instance to new versions supported by Amazon RDS. You can maintain compatibility with specific SQL Server versions, test new versions with your application before deploying in production, and perform version upgrades on your own terms and timelines.

Unless you specify otherwise, your DB Instance will automatically be upgraded to new SQL Server minor versions as they are supported by Amazon RDS. This patching will occur during your scheduled maintenance window, and it will be announced on the Amazon RDS Community Forum in advance. To turn off automatic version upgrades, set the **AutoMinorVersionUpgrade** parameter for your DB instance to *false*.

If you opt out of automatically scheduled upgrades, you can manually upgrade to a supported minor version release by following the same procedure as you would for a major version update. For information, see Major DB Engine Version Upgrades for a DB Instance (p. 447).

> **Note**
> Amazon RDS periodically aggregates official Microsoft SQL Server database patches and assigns an Amazon RDS-specific DB Engine version. The current supported versions are SQL Server 2008 R2 Service Pack 1 and SQL Server 2012.

Major Version Change: Upgrading from 2008 R2 to 2012

Amazon RDS supports major version upgrades from Microsoft SQL Server 2008 R2 to SQL Server 2012. You perform the upgrade by using the Amazon RDS modify DB instance operation. You should thoroughly test any major version upgrade before upgrading your production instances. For information about upgrading a DB instance, see Major DB Engine Version Upgrades for a DB Instance (p. 447)

Supported SQL Server Roles and Permissions

The SQL Server database engine uses role-based security. The master user name you use when you create a DB instance is a SQL Server Authentication login that is a member of the `processadmin`, `public`, and `setupadmin` fixed server roles.

Any user who creates a database will be assigned to the db_owner role for that database and will have all database-level permissions except for those that are used for backups. Amazon RDS manages backups for you.

The following server-level roles are not currently available in Amazon RDS:

- bulkadmin
- dbcreator
- diskadmin
- securityadmin
- serveradmin
- sysadmin

The following server-level permissions are not available on a SQL Server DB instance:

- ADMINISTER BULK OPERATIONS
- ALTER ANY CREDENTIAL
- ALTER ANY EVENT NOTIFICATION
- ALTER ANY EVENT SESSION
- ALTER ANY SERVER AUDIT
- ALTER RESOURCES
- ALTER SETTINGS (You can use the DB Parameter Group APIs to modify parameters. For more information, see Working with DB Parameter Groups (p. 523).
- AUTHENTICATE SERVER
- CONTROL_SERVER
- CREATE DDL EVENT NOTIFICATION
- CREATE ENDPOINT
- CREATE TRACE EVENT NOTIFICATION
- EXTERNAL ACCESS ASSEMBLY
- SHUTDOWN (You can use the RDS reboot option instead)
- UNSAFE ASSEMBLY
- ALTER ANY AVAILABILITY GROUP (SQL Server 2012 only)
- CREATE ANY AVAILABILITY GROUP (SQL Server 2012 only)

Using SSL with a SQL Server DB Instance

You can use SSL to encrypt connections between your applications and your Amazon RDS SQL Server DB instances. SSL support is available in all AWS regions for all supported SQL Server editions. Amazon RDS creates an SSL certificate for your SQL Server DB instance when the instance is created. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks.

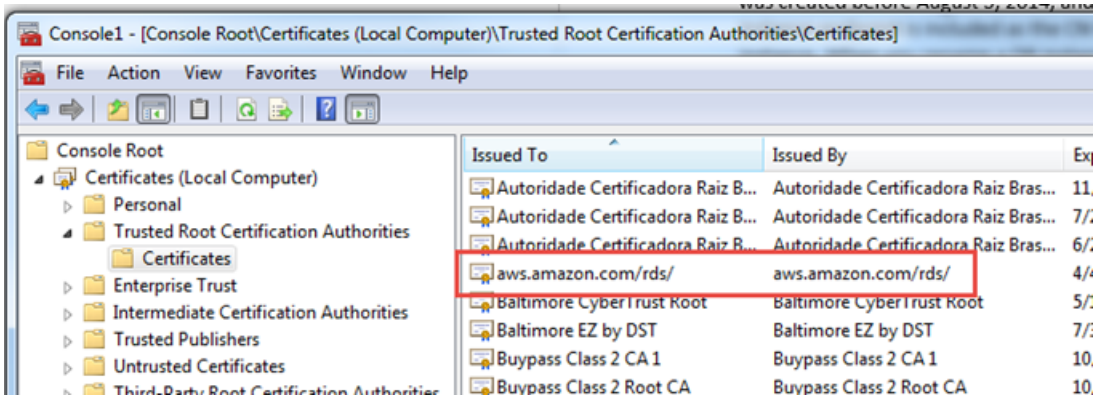
Important
Amazon RDS will rotate all SSL certificates for DB instances on March 23, 2015 but will not initiate a reboot of the instance. If you use SSL to connect to an Amazon RDS DB instance, you

must follow the steps in the topic SSL Certificate Rotation (p. 109) to apply a new SSL certificate to your DB instance before March 23, 2015 or you will not be able to connect to the DB instance using SSL.

Note that all new SQL Server instances created after August 5, 2014 will use the DB instance endpoint in the Common Name (CN) field of the SSL certificate. Prior to August 5, 2014, SSL certificate verification was not available for VPC based SQL Server instances. If you have a VPC based SQL Server DB instance that was created before August 5, 2014, and you want to use SSL certificate verification and ensure that the instance endpoint is included as the CN for the SSL certificate for that DB instance, then rename the instance. When you rename a DB instance, a new certificate is deployed and the instance is rebooted to enable the new certificate.

To encrypt connections to an Amazon RDS SQL Server DB instance using SSL, perform these steps on the client computer:

1. Download the certificate from http://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem.
2. Import the certificate into your Windows operating system:
    1. On the **Start** menu, type **Run** in the search box and hit **Enter**.
    2. In the **Open** box, type **MMC** and click **OK**.
    3. In the MMC console, on the **File** menu, click **Add/Remove Snap-in**.
    4. In the **Add or Remove Snap-ins** dialog box, select **Certificates** in the **Available snap-ins** box and click **Add**.
    5. In the MMC console, on the **File** menu, click **Add/Remove Snap-in**.
    6. In the **Certificates snap-in** dialog box, click **Computer account**, and then click **Next**.
    7. In the **Select computer** dialog box, click **Finish**.
    8. In the **Add or Remove Snap-ins** dialog box, click **OK**.
    9. In the MMC console, expand **Certificates**, right-click **Trusted Root Certification Authorities**, click **All Tasks**, and then click **Import**.
    10. On the Certificate Import Wizard first screen, click **Next**.
    11. On the Certificate Import Wizard second screen, click **Browse** and locate the rds-ssl-combined-ca-bundle.pem file you downloaded in step 1. You must change the file type in the browse window to **All files (*.*)** to do this, because .pem is not a standard certificate extension. Click **Open** to select the certificate file and then click **Next** in the wizard.
    12. On the Certificate Import Wizard third screen, click **Next**.
    13. On the Certificate Import Wizard fourth screen, click **Finish**. You should see a dialog box indicating that the import was successful.
    14. In the MMC console, expand **Certificates**, expand **Trusted Root Certification Authorities**, click **Certificates**, and locate the certificate to confirm it exists:



15. Restart the computer.

For more information about adding a certificate to a computer, go to the Windows documentation.

3. Connect to the Amazon RDS SQL DB instance.

   - In SQL Server Management Studio, follow these steps:
     1. Launch SQL Server Management Studio.
     2. In the **Connect to server** dialog box, enter the server information, login user name, and password.
     3. Click **Options>>**.
     4. Select **Encrypt connection**.
     5. Click **Connect**.

     For more information on SQL Server Management Studio, go to Use SQL Server Management Studio.
   - For any other SQL client, append "encrypt=true" to your connection string. This may be available as an option or property on the connection page in GUI tools.

     Note
     To enable SSL encryption for clients that connect using JDBC, you may need to add the Amazon RDS SQL certificate to the Java CA certificate (cacerts) store. You can do this by using the keytool utility.

4. Confirm the encrypted status of your connection by running the following query and verifying that `encrypt_option` is true:

```
SELECT encrypt_option FROM sys.dm_exec_connections WHERE session_id = @@SPID
```

Using the TDE Option to Encrypt Data at Rest

Most Amazon RDS DB engines support option groups that allow you to select additional features for your DB instance. SQL Server support includes the TDE option, which transparently encrypts stored data for SQL Server 2008 R2 Enterprise Edition and SQL Server 2012 Enterprise Edition. For more information about SQL Server TDE, see SQL Server Transparent Data Encryption (p. 350). For more information about working with option groups, see Working with Option Groups (p. 510).

Creating a DB Instance Running the SQL Server Database Engine

The basic building block of Amazon RDS is the DB instance. This is the environment where you run your SQL Server databases.

Important
You must complete the tasks in the Setting Up for Amazon RDS (p. 7) section before you can create or connect to a DB instance.

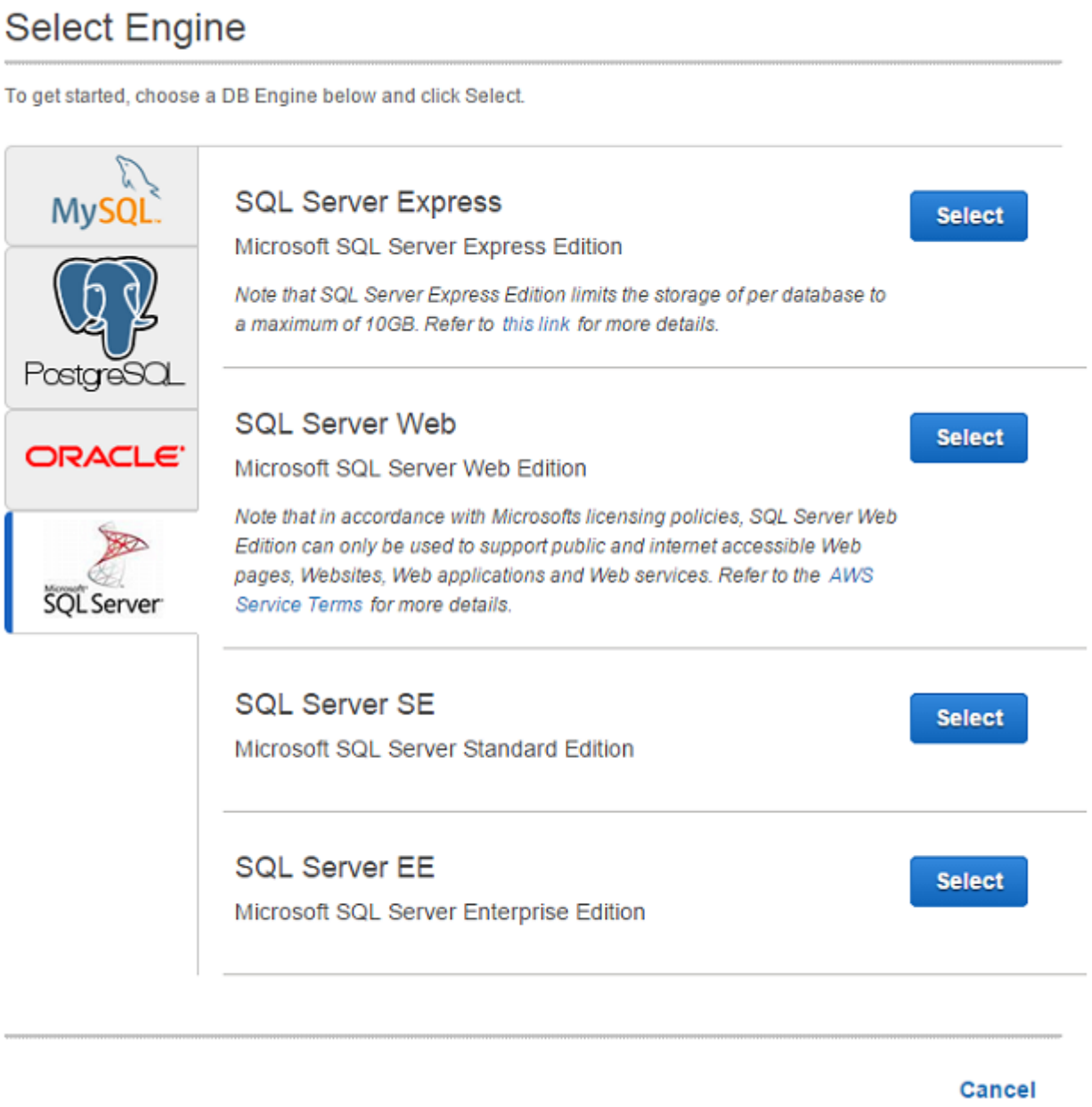
AWS Management Console

To create a DB Instance running the Microsoft SQL Server database engine

To launch a SQL Server DB instance

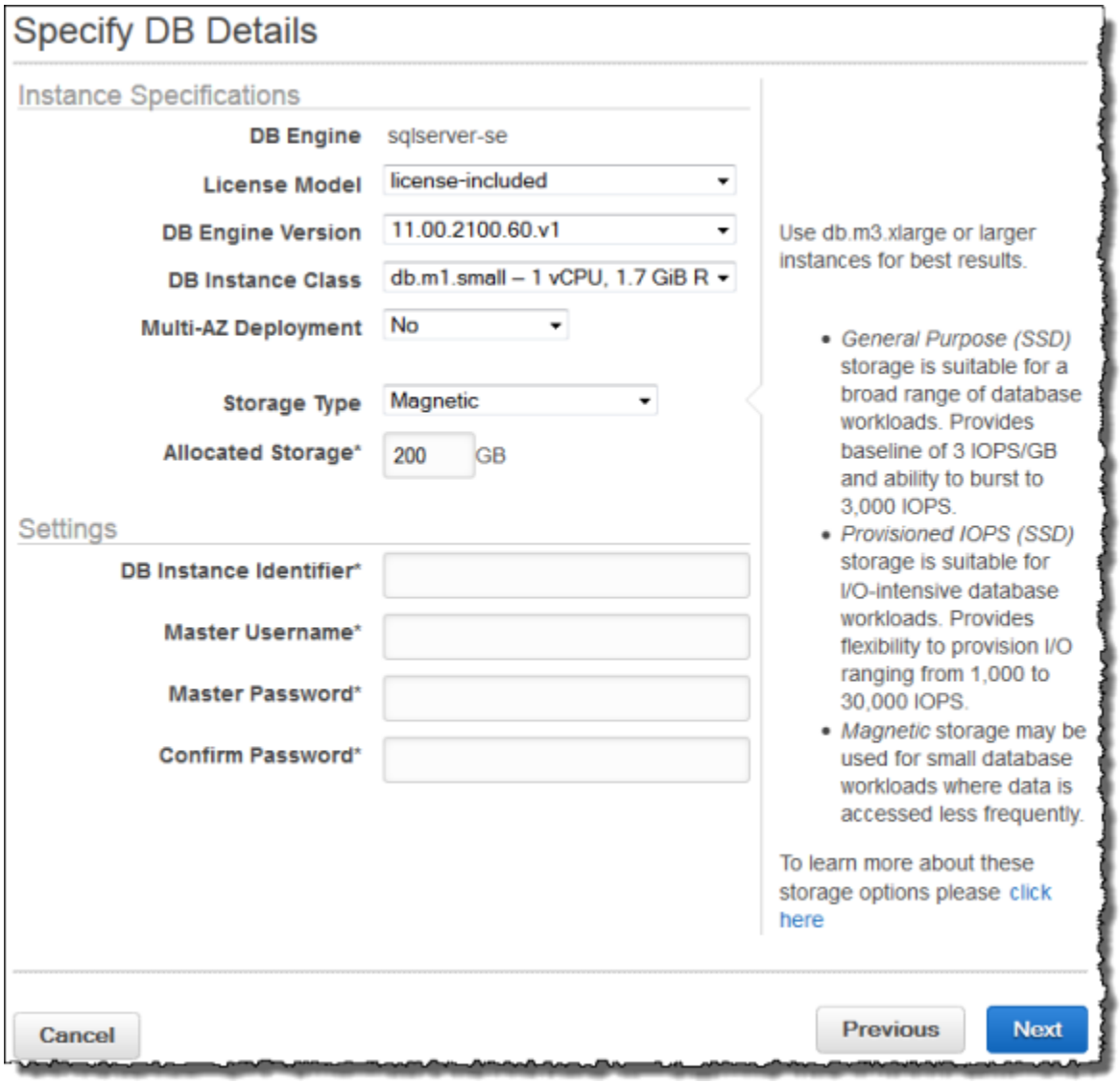
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the top right corner of the Amazon RDS console, select the region in which you want to create the DB instance.
3. In the navigation pane, click **DB Instances**.
4. Click **Launch DB Instance** to start the **Launch DB Instance Wizard**.

   The wizard opens on the **Select Engine** page. The SQL Server editions available will vary by region.


Select Engine
To get started, choose a DB Engine below and click Select.
MySQL
PostgreSQL
ORACLE
Microsoft SQL Server
SQL Server Express
Microsoft SQL Server Express Edition
Note that SQL Server Express Edition limits the storage of per database to a maximum of 10GB. Refer to this link for more details.
Select
SQL Server Web
Microsoft SQL Server Web Edition
Note that in accordance with Microsofts licensing policies, SQL Server Web Edition can only be used to support public and internet accessible Web pages, Websites, Web applications and Web services. Refer to the AWS Service Terms for more details.
Select
SQL Server SE
Microsoft SQL Server Standard Edition
Select
SQL Server EE
Microsoft SQL Server Enterprise Edition
Select
Cancel

5. In the **Select Engine** window, click the SQL Server icon and then click the **Select** button for the SQL Server DB engine edition you want to use.
6. The **Production?** step asks if you are planning to use the DB instance you are creating for production. If you are, select **Yes**. By selecting **Yes**, the failover option **Multi-AZ** and the **Provisioned IOPS** storage option will be preselected in the following step. These features are recommended for any production environment. Click **Next Step** when you are finished.
7. On the **Specify DB Details** page, specify your DB instance information. Click **Next** when you are finished.

For this parameter...	...Do this:
License Model	Select the licensing model you want to use. Select **`license-included`** to use the general license agreement for Microsoft SQL Server that is included with your DB instance, or select **`bring your own license`** to use your existing license. Each licensing model may not be available for all editions or in all regions.
DB Engine Version	Select the version of Microsoft SQL Server you want to use.
DB Instance Class	Select a configuration for your DB instance. For example, a **`db.m1.small`** instance class equates to 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity. If possible, select an instance class large enough that a typical query working set can be held in memory. This improves performance by allowing the system to avoid writing to disk. For more information about all the DB instance class options, see DB Instance Class (p. 65).
Multi-AZ Deployment	Select **`No`** to create your DB instance in a single availability zone, or select Yes to have a standby mirror of your DB instance created in another Availability Zone for failover support. For more information about multiple availability zones, see Regions and Availability Zones (p. 69).
Allocated Storage	Type a value to allocate storage for your DB instance (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon RDS Storage Types (p. 77).
Storage Type	Select the storage type you want to use. For more information about storage, see Storage for Amazon RDS (p. 77).
DB Instance Identifier	Type a name for the DB instance of 15 alphanumeric characters or less that is unique for your account in the region you selected. You may chose to add some intelligence to the name such as including the region and DB Engine you selected, such as **`sqlsv-instance1`**.
Master Username	Type a name that you will use as the master username to log on to your DB Instance with all database privileges. The master username is a SQL Server Authentication login that is a member of the `processadmin`, `public`, and `setupadmin` fixed server roles.
Master User Password and **Confirm Password**	Type a password that contains from 8 to 128 printable ASCII characters (excluding /,", a space, and @) for your master user password. Retype the password in the **Confirm Password** text box.


Specify DB Details
Instance Specifications
DB Engine
sqlserver-se
License Model
license-included
DB Engine Version
11.00.2100.60.v1
DB Instance Class
db.m1.small – 1 vCPU, 1.7 GiB R
Multi-AZ Deployment
No
Storage Type
Magnetic
Allocated Storage*
200
GB
Settings
DB Instance Identifier*
Master Username*
Master Password*
Confirm Password*
Use db.m3.xlarge or larger instances for best results.
General Purpose (SSD) storage is suitable for a broad range of database workloads. Provides baseline of 3 IOPS/GB and ability to burst to 3,000 IOPS.
Provisioned IOPS (SSD) storage is suitable for I/O-intensive database workloads. Provides flexibility to provision I/O ranging from 1,000 to 30,000 IOPS.
Magnetic storage may be used for small database workloads where data is accessed less frequently.
To learn more about these storage options please click here
Cancel
Previous
Next

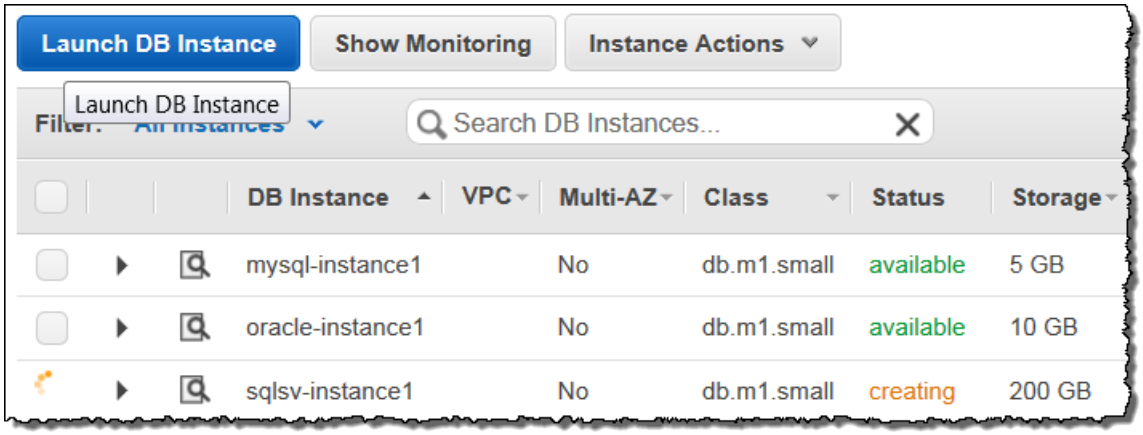
8. On the **Configure Advanced Settings** page, provide additional information that Amazon RDS needs to launch the SQL Server DB instance. Specify your DB instance information, then click Launch DB Instance.

For this parameter...	...Do this:
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, select the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, select **`Not in VPC`**. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).

For this parameter...	...Do this:
DB Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, select **`default`**, which will be the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, select the DB subnet group you created for that VPC. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).
Publicly Accessible	Select **`Yes`** to give the DB instance a public IP address, meaning that it will be accessible outside the VPC; otherwise, select **`No`**, so the DB instance will only be accessible from inside the VPC. For more information about hiding DB instances from public access, see Hiding a DB instance in a VPC from the Internet.
Availability Zone	Use the default value of **`No Preference`** unless you want to specify an Availability Zone.
VPC Security Group	If you are a new customer to AWS, select the default VPC. Otherwise, select the VPC security group you previously created.
Database Port	Specify a port you want to access the database through. SQL Server installations default to port 1433. If you use a DB security group with your DB instance, this must be the same port value you provided when creating the DB security group. **Important** You cannot change the port once you create the DB instance, so it is very important that you determine the correct port to use to access the DB instance.
Parameter Group	Select a DB parameter group. You can choose the default parameter group or you can create a parameter group and select that parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 523).
Option Group	Select an option group. You can choose the default option group or you can create an option group and select that option group. For more information about option groups, see Working with Option Groups (p. 510).
Enable Encryption	Select `Yes` to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 106).
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For non-trivial instances, set this value to **`1`** or greater.
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of **`No Preference`**.

For this parameter...	...Do this:
Auto Minor Version Upgrade	Select **`Yes`** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Maintenance Window	Select the 30 minute window in which pending modifications to your DB instance are applied. If you the time period doesn't matter, select **`No Preference`**.

9. On the final page of the wizard, click **Close**.
10. On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new instance to be available.



CLI

To create a DB Instance Running the Microsoft SQL Server Database Engine

- Use the command `rds-create-db-instance` to create a DB Instance.

```
PROMPT>rds-create-db-instance mymsftsqlserver -s 250 -c db.m1.large -e
sqlserver-se
   - u <masterawsuser> -p <masteruserpassword> --backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE  mymsftsqlserver  db.m1.large  sqlserver-se  250  sa  creating
 3  ****  n  10.50.2789
SECGROUP  default  active
PARAMGRP  default.sqlserver-se-10.5  in-sync
```

API

To create a DB Instance

- Call the `CreateDBInstance` action. For example, you could use the following parameters:
  - *`DBInstanceIdentifier`* = `mymsftsqlserver`
  - *`Engine`* = `sqlserver-se`
  - *`DBInstanceClass`* = `db.m1.large`

- *AllocatedStorage* = 250
- *BackupRetentionPeriod* = 3
- *MasterUsername* = <masterawsuser>
- *MasterUserPassword* = <masteruserpassword>

Example

```
https://rds.amazonaws.com/
   ?Action=CreateDBInstance
   &AllocatedStorage=250
   &BackupRetentionPeriod=3
   &DBInstanceClass=db.m1.large
   &DBInstanceIdentifier=mymsftsqlserver
   &DBName=mydatabase
   &DBSecurityGroups.member.1=mysecuritygroup
   &DBSubnetGroup=mydbsubnetgroup
   &Engine=sqlserver-se
   &MasterUserPassword=<masteruserpassword>
   &MasterUsername=<masterawsuser>
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140305/us-west-2/rds/aws4_request
   &X-Amz-Date=20140305T185838Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=b441901545441d3c7a48f63b5b1522c5b2b37c137500c93c45e209d4b3a064a3
```

Related Topics

- Amazon RDS DB Instances (p. 64)
- Amazon RDS Security Groups (p. 110)
- Connecting to a DB Instance Running the SQL Server Database Engine (p. 321)
- DB Instance Class (p. 65)
- Deleting a DB Instance (p. 459)

Connecting to a DB Instance Running the SQL Server Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. In order for you to connect, the DB instance must be associated with a security group containing the IP addresses and network configuration that you will use to access the DB instance. You may have already done this when you created the instance. If you assigned a default, non-configured security group when you created the instance, the DB instance firewall will prevent connections.

If you need to create a new security group to enable access, the type of security group you create will depend on what EC2 platform your DB instance is on, and whether you will be accessing your DB instance from an EC2 instance. For more information about the two EC2 platforms supported by Amazon RDS, *EC2-VPC* and *EC2-Classic*, see Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 563). In general, if your DB instance is on the *EC2-Classic* platform, you will need to create a DB security group; if your DB instance is on the *EC2-VPC* platform, you will need to create a VPC security group. For more information about security groups, see Amazon RDS Security Groups (p. 110).

Once you have created the security group, you must modify the DB instance to associate it with the security group. For more information on modifying the DB instance, see Modifying a DB Instance Running the SQL Server Database Engine (p. 328).

You can enhance security by using SSL to encrypt connections to the DB instance. For information on connecting to a DB instance using SSL, see Using SSL with a SQL Server DB Instance (p. 310).

The following examples assume that your DB instance has an appropriate security group.

Connecting with SQL Server Management Studio

This example shows how to connect to a DB instance running the Microsoft SQL Server database engine by using the Microsoft SQL Server Management Studio utility. For more information on using Microsoft SQL Server, go to the Microsoft SQL Server website.

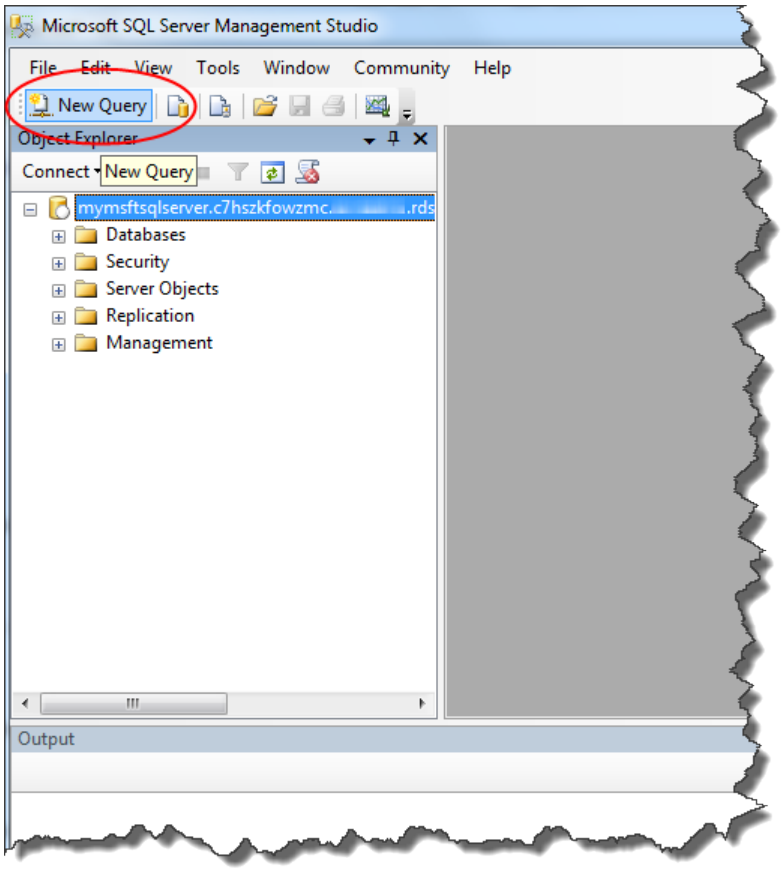
> **Note**
> This example uses the Microsoft SQL Server Management Studio utility. This utility is part of the Microsoft SQL Server software distribution. To download a stand-alone version of this utility, go to the Microsoft Download Center - Microsoft SQL Server Management Studio Express.

To connect to a DB instance using Microsoft SQL Server Management Studio

1. On the **Instances** page of the AWS Management Console, select the arrow next to the DB instance to show the instance details. Note the server name and port of the DB instance, which are displayed in the **Endpoint** field at the top of the panel, and the master user name, which is displayed in the **Username** field in the **Configuration Details** section. An example is shown following:

2. Open Microsoft SQL Server Management Studio. The **Connect to Server** dialog box appears, as shown following:

3. In the **Server type:** box, select **`Database Engine`**.
4. In the **Server name** box, type or paste the server name of the DB instance, type a comma ",", and then type the port number used by the DB instance. For example, the **Server name** value could be: **`sqlsvr-pdz.abcd12340.us-west-2.rds.amazonaws.com,1433`**.
5. In the **Authentication** box, select **`SQL Server Authentication`**.
6. In the **Login** box, type or paste the master user name for the DB instance.
7. In the **Password** box, type the password for the master user.
8. Click **Connect**. After a few moments, Microsoft SQL Server Management Studio should be connected to your DB instance.
9. Click **New Query** in the SQL Server Management Studio toolbar, as shown following:

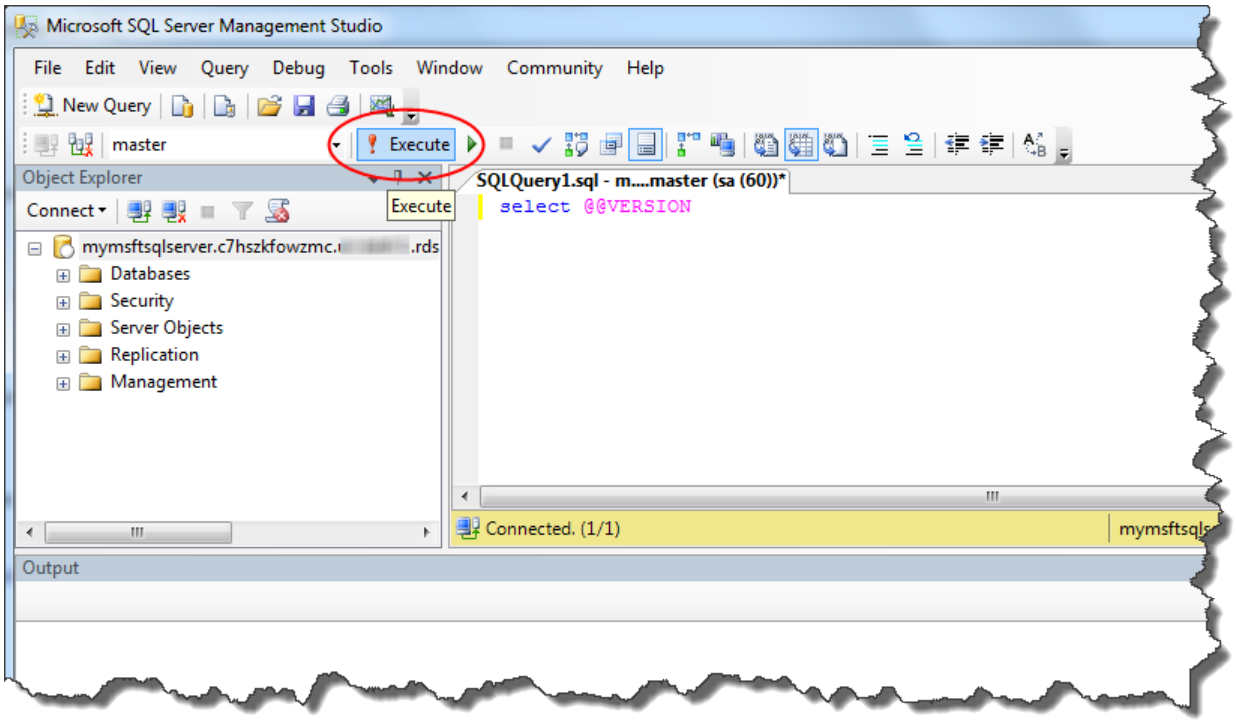


A new SQL query window will open.

10. Type the following SQL query:

```
select @@VERSION
```

11. Click **! Execute** on the SQL Enterprise Manager toolbar to run the query, as shown following:



The query should return the version information for your DB instance, similar to the following:

```
Microsoft SQL Server 2012 - 11.0.2100.60 (X64)
 Feb 10 2012 19:39:15
 Copyright (c) Microsoft Corporation
 Standard Edition (64-bit) on Windows NT 6.1 <X64> (Build 7601: Service Pack
 1) (Hypervisor)
```

Connecting with SQL Workbench/J

This example shows how to connect to a DB instance running the Microsoft SQL Server database engine by using the SQL Workbench/J database tool. This tool uses JDBC for the connection.

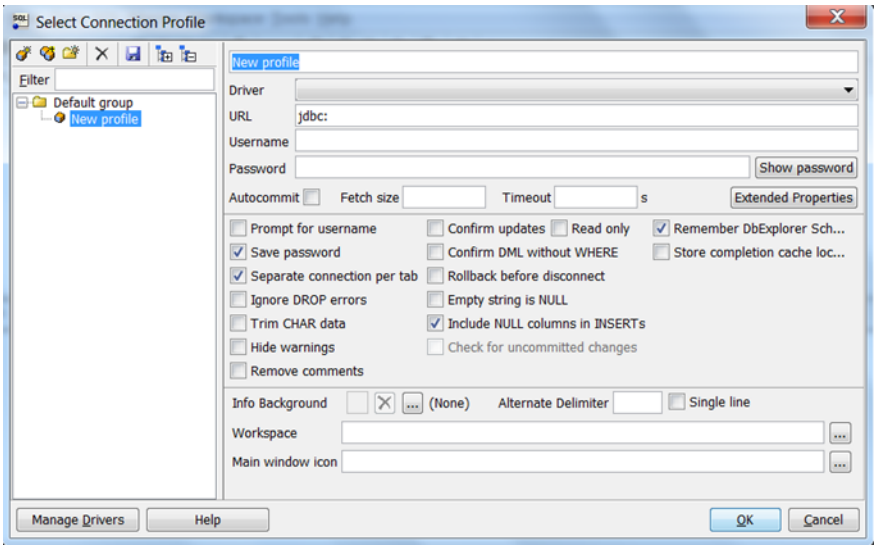
> **Note**
> This example uses the SQL Workbench/J database tool. To download this tool, go to the SQL Workbench/J website. It also requires the JDBC driver for SQL Server. To download this driver, go to Microsoft JDBC Drivers 4.1 (Preview) and 4.0 for SQL Server.

This example illustrates the minimal profile settings for making a connection. For more information on additional SQL Workbench/J profile settings, go to Connecting to the database in the SQL Workbench/J documentation.

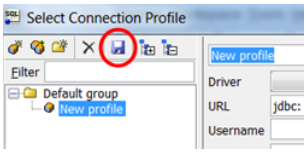
To connect to a DB instance using SQL Workbench/J

1. On the **Instances** page of the AWS Management Console, select the arrow next to the DB instance to show the instance details. Note the endpoint of the DB instance, which is displayed in the **Endpoint** field at the top of the panel, and the master user name, which is displayed in the **Username** field in the **Configuration Details** section. An example is shown following:

2. Open SQL Workbench/J. The **Select Connection Profile** dialog box appears, as shown following:



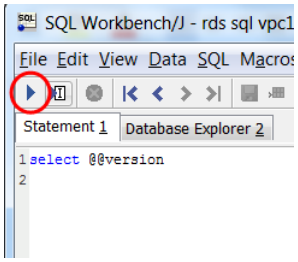
3. In the first box at the top of the dialog box, enter a name for the profile.
4. In the **Driver** box, select **`SQL JDBC 4.0`**.
5. In the **URL** box, type in **`jdbc:sqlserver://`**, then type or paste the endpoint used by the DB instance. For example, the **URL** value could be: **`jdbc:sqlserver://sqlsvr-pdz.abcd12340.us-west-2.rds.amazonaws.com:1433`**.
6. In the **Username** box, type or paste the master user name for the DB instance.
7. In the **Password** box, type the password for the master user.
8. Click the save icon in the dialog toolbar, as shown following:



9. Click **OK**. After a few moments, SQL Workbench/J should be connected to your DB instance.
10. In the query pane, type the following SQL query:

```
select @@VERSION
```

11. Click the execute icon in the toolbar, as shown following:



The query should return the version information for your DB instance, similar to the following:

```
Microsoft SQL Server 2012 - 11.0.2100.60 (X64)
```

Troubleshooting a Connection to a DB Instance Running SQL Server

There are several common causes for problems when trying to connect to a DB instance:

- The access rules enforced by your local firewall and the IP addresses you authorized to access your DB instance in the instance's security group are not in sync. The problem is most likely the egress or ingress rules on your firewall. For more information about security groups, see Amazon RDS Security Groups (p. 110).
- If you cannot send out or receive communications over the port you specified when you created the DB instance, you will not be able to connect to the DB instance. Check with your network administrator to determine if the port you specified for your DB instance is allowed to be used for inbound and outbound communication.
- For newly created DB instances, you must wait for the DB instance status to be "Available" before you can connect to the instance. Depending on the size of your DB instance, it can take up to 20 minutes before the instance is available.

SQL Server Management Studio Error Messages

Try the following solutions to common error messages from SQL Server Management Studio.

- **Could not open a connection to SQL Server - Microsoft SQL Server, Error: 53** - Make sure you included the port number when you specified the server name. For example, the server name for a DB instance (including the port number) could be: **`sqlsvr-pdz.abcd12340.region.rds.amazonaws.com,1433`**.
- **No connection could be made because the target machine actively refused it - Microsoft SQL Server, Error: 10061** - You were able to reach the DB instance but the connection was refused. This is often caused by the user name or password being incorrect.

Related Topics

- Amazon RDS DB Instances (p. 64)

- Creating a DB Instance Running the SQL Server Database Engine (p. 313)
- Amazon RDS Security Groups (p. 110)
- Deleting a DB Instance (p. 459)

Modifying a DB Instance Running the SQL Server Database Engine

You can change the settings of a DB instance to accomplish tasks such as changing the instance class or renaming the instance. This topic guides you through modifying an Amazon RDS SQL Server DB instance, and describes the settings for SQL Server instances. For information about additional tasks, such as renaming, rebooting, deleting, tagging, or upgrading an Amazon RDS DB instance, see Amazon RDS DB Instance Lifecycle (p. 440). We recommend that you test any changes on a test instance before modifying a production instance so you better understand the impact of a change. This is especially important when upgrading database versions.

Note
You cannot modify an existing SQL Server DB instance to change storage type or modify storage allocation.

You can have the changes apply immediately or have them applied during the DB instance's next maintenance window. Some modifications that do not cause a service interruption are applied immediately. Applying changes immediately can cause an interruption by restarting the DB instance in some cases; for more information on the impact of the **Apply Immediately** option when modifying a DB instance, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 455).

AWS Management Console

To modify an SQL Server DB Instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **DB Instances**.
3. Select the check box for the DB instance that you want to change, and then click **Modify**.
4. In the **Modify DB Instance** dialog box, change any of the following settings that you want:

Setting	Description
DB Engine Version	In the list provided, click the version of the SQL Server database engine that you want to use.
DB Instance Class	In the list provided, click the DB instance class that you want to use. For information about instance classes, see DB Instance Class (p. 65).
Multi-AZ Deployment	If you want to create a standby mirror of your DB instance in another Availability Zone, click **Yes**; otherwise, click **No**. For more information on Multi-AZ deployments using SQL Server Mirroring, see Planning your Multi-AZ Deployments Using SQL Server Mirroring (p. 307).
Allocated Storage	You cannot change the allocated storage for a SQL Server DB instance.
Storage Type	You cannot change the storage type for an existing SQL Server DB instance.

Setting	Description
DB Instance Identifier	You can rename the DB instance by typing a new name. When you change the DB instance identifier, an instance reboot will occur immediately if you set `Apply Immediately` to true, or will occur during the next maintenance window if you set `Apply Immediately` to false. This value is stored as a lowercase string.
New Master Password	Type a password that contains from 8 to 128 printable ASCII characters (excluding /,", a space, and @) for your master user password.
Security Group	Select the security group you want associated with the DB instance. For more information about security groups, see Working with DB Security Groups (p. 537).
Parameter Group	Select the parameter group you want associated with the DB instance. Changing this setting does not result in an outage. The parameter group name itself is changed immediately, but the actual parameter changes are not applied until you reboot the instance without failover. The DB instance will NOT be rebooted automatically and the parameter changes will NOT be applied during the next maintenance window. For more information about parameter groups, see Working with DB Parameter Groups (p. 523).
Option Group	Select the option group you want associated with the DB instance. For more information about option groups, see Working with Option Groups (p. 510).
Backup Retention Period	Specify the number of days that automatic backups will be retained. To disable automatic backups, set this value to 0. **Note** An immediate outage will occur if you change the backup retention period from 0 to a non-zero value or from a non-zero value to 0.
Backup Window	Set the time range during which automated backups of your databases will occur. Specify a start time in Universal Coordinated Time (UTC) and a duration in hours.
Auto Minor Version Upgrade	If you want your DB instance to receive minor engine version upgrades automatically when they become available, click **Yes**. Upgrades are installed only during your scheduled maintenance window.
Maintenance Window	Set the time range during which system maintenance, including upgrades, will occur. Specify a start time in UTC and a duration in hours.

5. To apply the changes immediately, select the **Apply Immediately** check box. Selecting this option can cause an outage in some cases; for more information on the impact of the **Apply Immediately** option, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 455).
6. When all the changes are as you want them, click **Yes, Modify**. If instead you want to cancel any changes that you didn't apply in the previous step, click **Cancel**.

CLI

To modify an SQL Server DB instance

- Use the command `rds-modify-db-instance`.

API

To modify an SQL Server DB instance

- Use the `ModifyDBInstance` `action`.

Working with SQL Server Multi-AZ with Mirroring

The simplest way to enable Multi-AZ for a SQL Server DB instance is to use the Amazon RDS console. When creating a new DB instance, you can simply select *Yes (Mirroring)* from the Multi-AZ drop down list in the **Launch DB Instance** wizard. You can also modify an existing SQL Server DB instance to use Multi-AZ; for information on modifying a DB instance, see Modifying a DB Instance Running the SQL Server Database Engine (p. 328).

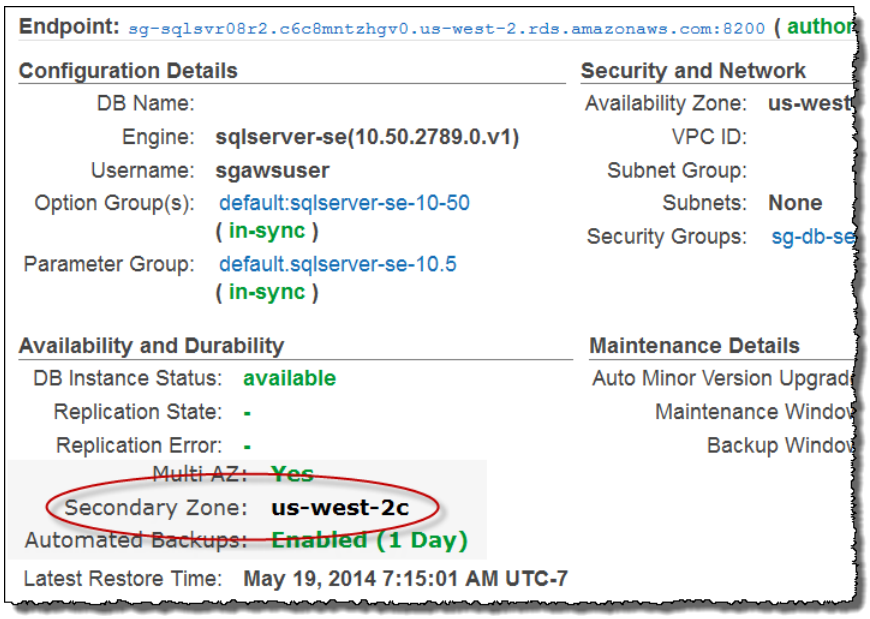
> **Note**
> SQL Server Multi-AZ using Mirroring is currently available in the US East (N. Virginia), US West (Oregon), and EU (Ireland) AWS regions. We plan to support other regions in the future.

To use SQL Server Mirroring with a SQL Server DB instance in a VPC, you must create a DB subnet group that has 3 subnets in distinct Availability Zones. You must then assign the DB subnet group to the SQL Server DB instance that is being mirrored.

When the DB instance is being modified for a Multi-AZ deployment, it will have a status of **Modifying**. During this phase, Amazon RDS creates the standby mirror, makes a backup of the primary DB instance, and updates the associated option group. Once the process is complete, the status of the primary DB instance becomes **Available**.

Determining the Location of the Standby Mirror

You can determine the location of the standby mirror by using the Amazon RDS console. You need to know the location of the standby mirror if you are setting up your primary DB instance in a VPC.



You can also view the Availability Zone of the standby mirror using the RDS CLI command `rds-describe-db-instances` or RDS API action `DescribeDBInstances`. The output will show the secondary AZ where the standby mirror is located.

Related Topics

- Planning your Multi-AZ Deployments Using SQL Server Mirroring (p. 307)

Importing and Exporting SQL Server Data

Use the following procedures to import data into and export data from an Amazon RDS SQL DB instance.

Topics

- Importing Data into SQL Server on Amazon RDS (p. 332)
- Exporting Data from SQL Server on Amazon RDS (p. 339)

Importing Data into SQL Server on Amazon RDS

If you have an existing Microsoft SQL Server deployment that you want to move to Amazon RDS, the complexity of your task depends on the size of your database and the types of database objects that you are transferring. For example, a database that contains data sets on the order of gigabytes, along with stored procedures and triggers, is going to be more complicated than a simple database with only a few megabytes of test data and no triggers or stored procedures.

RDS for SQL Server service does not currently support RESTORE DATABASE ... FROM FILE, because the database and log file backups must be local to the SQL Server instance. Similarly, FILESTREAM is also not supported at this time.

The BULK INSERT and OPENROWSET(BULK...) statements from the server are not supported import procedures due to their dependency on the ADMINISTER BULK OPERATIONS permission which is not granted for SQL Server DB instances. Please use the process outlined below to import data to a SQL Server DB instance.

The process that we recommend to import data into a SQL Server DB instance is as follows:

1. Create a DB Instance. (p. 313)
2. Before you load data into the destination DB Instance, you should do some preparation (p. 332), such as disabling foreign key constraints and database triggers. You should also disable automated backups.
3. Query the source SQL Server instance for any logins that you want to import (p. 334) to the destination DB Instance.
4. In your existing SQL Server deployment, generate scripts that obtain data from the source SQL Server instance, and then apply the scripts to the destination DB Instance (p. 335). If you have existing scripts, you can apply those scripts to the destination DB Instance. If you are importing a large dataset, your script can define only the database schema; otherwise, it can also include the data and all other database objects.
5. After your data is imported, reverse any preparations that you made earlier (p. 337), re-enable foreign key constraints and database triggers, switch the recovery model to its original state, and then re-enable automated backups.

Note
Amazon RDS for SQL Server does not currently support importing data into the msdb database, though we do support SQL Server Agent jobs. Some SQL Server features that use the msdb database, such as Database Mail and Replication, are not currently supported in Amazon RDS.

Preparing to Import Data into Your SQL Server DB Instance

Before you import data into your SQL Server DB Instance, we recommend the following best practices:

- Stop applications from accessing the destination DB Instance.
- Create a snapshot of the target database.

- Disable automated backups on the target database.
- Disable foreign key constraints, if applicable.
- Drop indexes, if applicable.
- Disable database triggers, if applicable.

Stop Applications from Accessing the Target DB Instance

If you prevent access to your DB Instance while you are importing data, data transfer will be faster. Additionally, you won't need to worry about conflicts while data is being loaded if other applications cannot write to the DB Instance at the same time. If something goes wrong and you have to roll back to a prior database snapshot, the only changes that you will lose will be the imported data, which you can import again after you resolve the issue.

For information about controlling access to your DB Instance, see Working with DB Security Groups (p. 537).

Create a Database Snapshot

If the target database is already populated with data, we recommend that you take a snapshot of the database before you import the data. If something goes wrong with the data import or you want to discard the changes, you can restore the database to its previous state by using the snapshot. For information about database snapshots, see Creating a DB Snapshot (p. 499).

Note
When you take a database snapshot, I/O operations to the database are suspended for about 10 seconds while the backup is in progress.

Disable Automated Backups

Disabling automated backups on the target DB Instance will improve performance while you are importing your data because Amazon RDS doesn't log transactions when automatic backups are disabled. There are, however, some things to consider. Because automated backups are required to perform a point-in-time recovery, you won't be able to restore the database to a specific point in time while you are importing data. Additionally, any automated backups that were created on the DB Instance are erased. You can still use previous snapshots to recover the database, and any snapshots that you have taken will remain available. For information about automated backups, see Working With Automated Backups (p. 496).

Disable Foreign Key Constraints

If you need to disable foreign key constraints, you can do so with the following script.

```
--Disable foreign keys on all tables
DECLARE @table_name SYSNAME;
DECLARE @cmd NVARCHAR(MAX);
DECLARE table_cursor CURSOR FOR SELECT name FROM sys.tables;

OPEN table_cursor;
FETCH NEXT FROM table_cursor INTO @table_name;

WHILE @@FETCH_STATUS = 0 BEGIN
  SELECT @cmd = 'ALTER TABLE '+QUOTENAME(@table_name)+' NOCHECK CONSTRAINT ALL';

  EXEC (@cmd);
  FETCH NEXT FROM table_cursor INTO @table_name;
```

```
END

CLOSE table_cursor;
DEALLOCATE table_cursor;

GO
```

Disable Database Triggers

If you need to disable database triggers, you can do so with the following script.

```
--Disable triggers on all tables
DECLARE @enable BIT = 0;
DECLARE @trigger SYSNAME;
DECLARE @table SYSNAME;
DECLARE @cmd NVARCHAR(MAX);
DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
 table_object.name table_name
FROM sysobjects trigger_object
JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
WHERE trigger_object.type = 'TR';

OPEN trigger_cursor;
FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

WHILE @@FETCH_STATUS = 0 BEGIN
  IF @enable = 1
     SET @cmd = 'ENABLE ';
  ELSE
     SET @cmd = 'DISABLE ';

  SET @cmd = @cmd + ' TRIGGER dbo.'+QUOTENAME(@trigger)+' ON dbo.'+QUOTE
NAME(@table)+' ';
  EXEC (@cmd);
  FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
END

CLOSE trigger_cursor;
DEALLOCATE trigger_cursor;

GO
```

Import Logins to Your SQL Server DB Instance

SQL Server stores logins and passwords in the `master` database. Because Amazon RDS does not grant access to the `master` database, you cannot directly import logins and passwords into your destination DB Instance. Instead, you must query the `master` database on the source SQL Server instance to generate a DDL file that includes all logins and passwords that you want to add to the destination DB Instance, as well as role memberships and permissions that you want to transfer.

For information about querying the `master` database, go to How to Transfer the Logins and the Passwords Between Instances of SQL Server 2005 and SQL Server 2008 on the Microsoft Knowledge Base.

The output of the script is another script that you can run on the destination DB Instance. Amazon RDS currently supports only SQL Server Authentication. Attempts to log in by using Windows Authentication will fail. You can ignore these failures, or you can edit the Microsoft script to include only logins that use SQL Server Authentication. Where the script in the Knowledge Base article has the following:

```
p.type IN
```

Use the following instead:

```
p.type = 'S'
```

Import the Data

Microsoft SQL Server Management Studio is a graphical SQL Server client that is included in all Microsoft SQL Server editions except the Express Edition. SQL Server Management Studio Express is available from Microsoft as a free download.

Note
SQL Server Management Studio is available only as a Windows-based application.

SQL Server Management Studio includes the following tools, which are useful in importing data to a SQL Server DB Instance:

- Generate and Publish Scripts Wizard
- Import and Export Wizard
- Bulk copy feature

Generate and Publish Scripts Wizard

The Generate and Publish Scripts Wizard creates a script that contains the schema of a database, the data itself, or both. If you generate a script for a database in your local SQL Server deployment, you can then run the script to transfer the information that it contains to an Amazon RDS DB Instance.

Note
For databases of 1 GB or larger, it is more efficient to script only the database schema and then use the Import and Export Wizard or the bulk copy feature of SQL Server to transfer the data.

For detailed information about the Generate and Publish Scripts Wizard, see the Microsoft SQL Server documentation.

In the wizard, pay particular attention to the advanced options on the **Set Scripting Options** page to ensure that everything you want your script to include is selected. For example, by default, database triggers are not included in the script.

When the script is generated and saved, you can use SQL Server Management Studio to connect to your DB Instance and then run the script.

Import and Export Wizard

The Import and Export Wizard creates a special Integration Services package, which you can use to copy data from your local SQL Server database to the destination DB Instance. The wizard can filter which tables and even which tuples within a table are copied to the destination DB Instance.

> **Note**
> The Import and Export Wizard works well for large datasets, but it may not be the fastest way to remotely export data from your local deployment. For an even faster way, you may want to consider the SQL Server bulk copy feature.

For detailed information about the Import and Export Wizard, go to the Microsoft SQL Server documentation

In the wizard, on the **Choose a Destination** page, do the following:

- In the **Server Name** box, enter the name of the endpoint for your DB Instance.
- For the server authentication mode, click **Use SQL Server Authentication**.
- Under **User name** and **Password**, enter the credentials for the master user that you created for the DB Instance.

Bulk Copy

The SQL Server bulk copy feature is an efficient means of copying data from a source database to your DB Instance. Bulk copy writes the data that you specify to a data file, such as an ASCII file. You can then run bulk copy again to write the contents of the file to the destination DB Instance.

This section uses the **bcp** utility, which is included with all editions of SQL Server. For detailed information about bulk import and export operations, go to the Microsoft SQL Server documentation.

> **Note**
> Before you use bulk copy, you must first import your database schema to the destination DB Instance. The Generate and Publish Scripts Wizard, described earlier in this topic, is an excellent tool for this purpose.

The following command connects to the local SQL Server instance to generate a tab-delimited file of a specified table in the C:\ root directory of your existing SQL Server deployment. The table is specified by its fully qualified name, and the text file has the same name as the table that is being copied.

```
PROMPT> bcp dbname.schema_name.table_name out C:\table_name.txt -n -S localhost
 -U username -P password -b 10000
```

Where:

- `-n` specifies that the bulk copy will use the native data types of the data to be copied.
- `-S` specifies the SQL Server instance that the *bcp* utility will connect to.
- `-U` specifies the user name of the account that will log in to the SQL Server instance.
- `-P` specifies the password for the user specified by `-U`.
- `-b` specifies the number of rows per batch of imported data.

> **Note**
> There may be other parameters, such as the `-E` parameter that pertains to identity values, that are important to your import situation; please review the full description of the command line syntax for the **bcp** utility, at the Microsoft SQL Server documentation.

For example, suppose a database named `store` that uses the default schema, `dbo`, contains a table named `customers`. The user account `admin`, with the password `insecure`, will copy 10,000 rows of the `customers` table to a file named `customers.txt`.

```
PROMPT> bcp store.dbo.customers out C:\customers.txt -n -S localhost -U admin
-P insecure -b 10000
```

After you generate the data file, if you have created the database and schema on the target DB Instance, you can upload the data to your DB Instance by using a similar command. In this case, you will use the `in` argument to specify an input file instead of `out` to specify an output file. Instead of using localhost to specify the local SQL Server instance, you will specify the endpoint of your DB Instance. If you use a port other than 1433, you will specify that, too. The user name and password will be those of the master user and password for your DB Instance. The syntax is as follows:

```
PROMPT> bcp dbname.schema_name.table_name in C:\table_name.txt -n -S end
point,port -U master_user_name -P master_user_password -b 10000
```

To continue the previous example, suppose the master user name is `admin`, and the password is `insecure`. The endpoint for the DB Instance is `rds.ckz2kqd4qsn1.us-east-1.rds.amazonaws.com`, and you will use port 4080. The command would be as follows:

```
PROMPT> bcp store.dbo.customers in C:\customers.txt -n -S rds.ckz2kqd4qsn1.us-
east-1.rds.amazonaws.com,4080 -U admin -P insecure -b 10000
```

Cleaning Up

If you followed the best practices outlined earlier in this topic for preparing to import data to your DB Instance, you will need to perform the following tasks now:

- Grant applications access to the target DB Instance.
- Enable automated backups on the target DB Instance.
- Enable foreign key constraints.
- Enable database triggers.

Grant Applications Access to the Target DB Instance

When your data import is complete, you can grant access to the DB Instance to those applications that you blocked during the import. For information about controlling access to your DB Instance, see Working with DB Security Groups (p. 537).

Enable Automated Backups on the Target DB Instance

For information about automated backups, see Working With Automated Backups (p. 496).

Enable Foreign Key Constraints

If you disabled foreign key constraints earlier, you can now enable them with the following script:

```
--Enable foreign keys on all tables
DECLARE @table_name SYSNAME;
DECLARE @cmd NVARCHAR(MAX);
DECLARE table_cursor CURSOR FOR SELECT name FROM sys.tables;

OPEN table_cursor;
FETCH NEXT FROM table_cursor INTO @table_name;

WHILE @@FETCH_STATUS = 0 BEGIN
  SELECT @cmd = 'ALTER TABLE '+QUOTENAME(@table_name)+' CHECK CONSTRAINT ALL';

  EXEC (@cmd);
  FETCH NEXT FROM table_cursor INTO @table_name;
END

CLOSE table_cursor;
DEALLOCATE table_cursor;
```

Enable Database Triggers

If you disabled database triggers earlier, you can now enable them with the following script:

```
--Enable triggers on all tables
DECLARE @enable BIT = 1;
DECLARE @trigger SYSNAME;
DECLARE @table SYSNAME;
DECLARE @cmd NVARCHAR(MAX);
DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
 table_object.name table_name
FROM sysobjects trigger_object
JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
WHERE trigger_object.type = 'TR';

OPEN trigger_cursor;
FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

WHILE @@FETCH_STATUS = 0 BEGIN
  IF @enable = 1
     SET @cmd = 'ENABLE ';
  ELSE
     SET @cmd = 'DISABLE ';

  SET @cmd = @cmd + ' TRIGGER dbo.'+QUOTENAME(@trigger)+' ON dbo.'+QUOTE
NAME(@table)+' ';
  EXEC (@cmd);
  FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
END

CLOSE trigger_cursor;
DEALLOCATE trigger_cursor;
```

Exporting Data from SQL Server on Amazon RDS

You can export data from an Amazon RDS SQL DB instance using either of a couple of options, following. Which option you choose depends on the target database and what you want to export.

- SQL Server Import and Export Wizard (p. 339) – You can use this wizard to copy one or more tables, views, or queries from your Amazon RDS SQL DB instance to another data store. The wizard can both export the data from your SQL Server DB instance and import it into the target data store. This choice is best if you want to transfer small- to medium-size tables or query result sets to another SQL Server DB instance, or if the target data store is not SQL Server.
- SQL Server Generate and Publish Scripts Wizard and bcp Utility (p. 340) – You can use this wizard to create scripts for an entire database or just selected objects. You can run these scripts on a target SQL Server DB instance to recreate the scripted objects. You can then use the bcp utility to bulk export the data for the selected objects to the target DB instance. This choice is best if you want to move a whole database (including objects other than tables) or large quantities of data between two SQL Server DB instances.

The tools named preceding are available as part of Microsoft SQL Server Management Studio, a graphical SQL Server client that is included in all Microsoft SQL Server editions except the Express Edition. SQL Server Management Studio Express is also available from Microsoft as a free download.

Note
SQL Server Management Studio is available only as a Windows-based application.

SQL Server Import and Export Wizard

To use the SQL Server Import and Export Wizard to export data, follow these steps:

1. In SQL Server Management Studio, connect to your Amazon RDS SQL DB instance. For details on how to do this, see Connecting to a DB Instance Running the SQL Server Database Engine (p. 321).
2. In **Object Explorer**, expand **Databases**, right-click the source database, select **Tasks**, and then click **Export Data** to open the wizard.
3. On the **Choose a Data Source** page, do the following:
   a. Click **`SQL Server Native Client 11.0`** in the **Data source** box.
   b. Verify that the **Server name** box shows the endpoint of your Amazon RDS SQL DB instance.
   c. Click **Use SQL Server Authentication**. Type the master user name and password of your Amazon RDS SQL DB instance in the **User name** and **Password** boxes.
   d. Verify that the **Database** box shows the database from which you want to export data.
   e. Click **Next**.
4. On the **Choose a Destination** page, do the following:
   a. Click **`SQL Server Native Client 11.0`** in the **Destination** box.

      Note
      Other target data sources are available, include .NET Framework data providers, OLE DB providers, SQL Server Native Client providers, ADO.NET providers, Microsoft Office Excel, Microsoft Office Access, and the Flat File source. If you choose to target one of these data sources, skip the remainder of step 4 and go to Choose a Destination in the SQL Server documentation for details on the connection information to provide.

   b. Type the server name of the target SQL Server DB instance in the **Server name** box.
   c. Click the appropriate authentication type. Type a user name and password if necessary.
   d. Click the database name of the target database in the **Database** box, or click **New** to create a new database to contain the exported data.

If you click **New**, go to Create Database in the SQL Server documentation for details on the database information to provide.

e. Click **Next**.

5. On the **Table Copy or Query** page, click **Copy data from one or more tables or views** or **Write a query to specify the data to transfer**. Click **Next**.
6. If you clicked **Write a query to specify the data to transfer**, you see the **Provide a Source Query** page. Type or paste in a SQL query, and then click **Parse** to verify it. Once the query validates, click **Next**.
7. On the **Select Source Tables and Views** page, do the following:
   a. Select the tables and views that you want to export, or verify that the query you provided is selected.
   b. Click **Edit Mappings** and specify database and column mapping information. For further details, go to Column Mappings in the SQL Server documentation.
   c. (Optional) To see a preview of data to be exported, select the table, view, or query, and then click **Preview**.
   d. Click **Next**.
8. On the **Run Package** page, verify that **Run immediately** is selected. Click **Next**.
9. On the **Complete the Wizard** page, verify that the data export details are as you expect. Click **Finish**.
10. On the **The execution was successful** page, click **Close**.

For more information, go to SQL Server Import and Export Wizard in the SQL Server documentation.

SQL Server Generate and Publish Scripts Wizard and bcp Utility

To use the SQL Server Generate and Publish Scripts Wizard and the bcp utility to export data, follow these steps:

1. In SQL Server Management Studio, connect to your Amazon RDS SQL DB instance. For details on how to do this, see Connecting to a DB Instance Running the SQL Server Database Engine (p. 321).
2. In **Object Explorer**, expand the **Databases** node and select the database you want to script.
3. Follow the instructions in Generate and Publish Scripts Wizard in the SQL Server documentation to create a script file.
4. In SQL Server Management Studio, connect to your target SQL Server DB instance.
5. With the target SQL Server DB instance selected in **Object Explorer**, click **Open** on the **File** menu, click **File**, and then open the script file.
6. If you have scripted the entire database, review the CREATE DATABASE statement in the script to make sure the database is being created in the location and with the parameters that you want. For further details, go to CREATE DATABASE in the SQL Server documentation.
7. If you are creating database users in the script, check to see if server logins exist on the target DB instance for those users. If not, create logins for those users; the scripted commands to create the database users will fail otherwise. For more information, go to Create a Login in the SQL Server documentation.
8. Click **!Execute** on the SQL Editor menu to execute the script file and create the database objects. When the script finishes, verify that all database objects exist as expected.
9. Use the bcp utility to export data from the Amazon RDS SQL DB instance into files. Open a command prompt and type the command

```
C:\> bcp database_name.schema_name.table_name out data_file -n -S
aws_rds_sql_endpoint -U username -P password
```

where:

- *table_name* is the name of one of the tables that you've re-created in the target database and now want to populate with data.
- *data_file* is the full path and name of the data file to be created.
- `-n` specifies that the bulk copy will use the native data types of the data to be copied.
- `-S` specifies the SQL Server DB instance to export from.
- `-U` specifies the user name to use when connecting to the SQL Server DB instance.
- `-P` specifies the password for the user specified by `-U`.

The following shows an example command:

```
bcp world.dbo.city out C:\Users\JohnDoe\city.dat -n -S sql-jdoe.1234abcd.us-
west-2.rds.amazonaws.com,1433 -U JohnDoe -P ClearTextPassword
```

For a full description of the bcp command line syntax, go to bcp Utility in the Microsoft SQL Server documentation.

Repeat this step until you have data files for all of the tables you want to export.

10. Prepare your target DB instance for bulk import of data by following the instructions at Basic Guidelines for Bulk Importing Data in the SQL Server documentation.
11. Decide on a bulk import method to use after considering performance and other concerns discussed in About Bulk Import and Bulk Export Operations in the SQL Server documentation.
12. Bulk import the data from the data files you created using the bcp utility, following the instructions at either Import and Export Bulk Data by Using the bcp Utility or Import Bulk Data by Using BULK INSERT or OPENROWSET(BULK...) in the SQL Server documentation, depending on what you decided in step 11.

Appendix: Common DBA Tasks for SQL Server

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB Instances that are running the Microsoft SQL Server database engine. In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB Instances, and it restricts access to certain system procedures and tables that require advanced privileges.

> **Note**
> When working with a SQL Server DB Instance, you can run scripts to modify a newly created database, but you cannot modify the [model] database, the database used as the model for new databases.

For information on working with SQL Server log files on Amazon RDS, see SQL Server Database Log Files (p. 602).

Topics

- Determining a Recovery Model (p. 342)
- Collations and Character Sets for SQL Server (p. 342)
- Resetting the db_owner Role Password (p. 343)
- Transitioning a Database from OFFLINE to ONLINE (p. 343)
- Dropping a Database in a Multi-AZ Deployment Using Mirroring (p. 343)
- Analyzing Your Database Workload on a DB Instance Using SQL Server Tuning Advisor (p. 344)
- Using SQL Server Agent (p. 346)
- Working with SQL Server Logs (p. 348)
- Handling UTC Times for Time Zone Awareness (p. 348)

Determining a Recovery Model

In RDS, the recovery model, retention period, and database status are linked. Changes to one can impact the other settings. For example:

- Changing a database's recovery model to "Simple" while backup retention is enabled will result in RDS setting the recovery model to "Full" within five minutes of the setting change. This will also result in Amazon RDS taking a snapshot of the DB instance.
- Setting the backup retention to "0" days results in RDS setting the recovery mode to "Simple."
- Changing a database's recovery model from "Simple" to any other option while backup retention is set to "0" days results in RDS setting the recovery model back to "Simple."

Collations and Character Sets for SQL Server

Amazon RDS creates a default server collation for character sets when a SQL Server DB instance is created. This default server collation is currently English (United States), or more precisely, SQL_Latin1_General_CP1_CI_AS. You can change the default collation at the database, table, or column level by overriding the collation when creating a new database or database object. For example, you can change from the default collation SQL_Latin1_General_CP1_CI_AS to Japanese_CI_AS for Japanese collation support. Even arguments in a query can be type-cast to use a different collation if necessary.

For example, the following query would change the default collation for the newly created database to Japanese_CI_AS:

```
CREATE TABLE [dbo].[Account]
(
    [AccountID] [nvarchar](10) NOT NULL,
    [AccountName] [nvarchar](100) COLLATE Japanese_CI_AS NOT NULL
) ON [PRIMARY];
```

The SQL Server DB engine supports Unicode by the built-in NCHAR, NVARCHAR, and NTEXT data types. For example, if you need CJK support, use these Unicode data types for character storage and override the default server collation when creating your databases and tables. Here are several links from Microsoft covering collation and Unicode support for SQL Server:

- Working with Collations
- Collation and International Terminology
- Using SQL Server Collations
- International Considerations for Databases and Database Engine Applications

Resetting the db_owner Role Password

If you lock yourself out of the db_owner role on your SQL database, you can reset the db_owner role password by modifying the DB instance password. By changing the DB instance password, you can regain access to the DB instance, access databases using the modified password for the db_owner, and restore privileges for the db_owner role that may have been accidentally revoked. You can change the DB instance password by using the Amazon RDS console, the Amazon RDS CLI command rds-modify-db-instance, or by using the ModifyDBInstance action. For more information about modifying a SQL Server DB instance, see Modifying a DB Instance Running the SQL Server Database Engine (p. 328).

Transitioning a Database from OFFLINE to ONLINE

SQL Server method	Amazon RDS method
ALTER DATABASE *name* SET ONLINE;	EXEC rdsadmin.dbo.rds_set_database_online *name*

Dropping a Database in a Multi-AZ Deployment Using Mirroring

If you need to drop a SQL Server database that is on a DB instance in a Multi-AZ deployment using Mirroring, you can use the following commands:

```
ALTER DATABASE <database_name> SET PARTNER OFF;
GO
DROP DATABASE <database_name>;
GO
```

Analyzing Your Database Workload on a DB Instance Using SQL Server Tuning Advisor

The Database Engine Tuning Advisor is a client application provided by Microsoft that analyzes database workload and recommends an optimal set of indexes for your SQL Server databases based on the kinds of queries you run. Like SQL Server Management Studio, you run Tuning Advisor from a client computer that connects to your RDS DB Instance that is running SQL Server. The client computer can be a local computer that you run on premises within your own network or it can be an Amazon EC2 Windows instance that is running in the same region as your RDS DB Instance.

This section shows how to capture a workload for Tuning Advisor to analyze. This is the preferred process for capturing a workload because RDS restricts host access to the SQL Server instance. The full documentation on Tuning Advisor can be found on MSDN.

To use Tuning Advisor, you must provide what is called a workload to the advisor. A workload is a set of Transact-SQL statements that execute against a database or databases that you want to tune. Database Engine Tuning Advisor uses trace files, trace tables, Transact-SQL scripts, or XML files as workload input when tuning databases. When working with RDS, a workload can be a file on a client computer or a database table on an RDS SQL Server DB accessible to your client computer. The file or the table must contain queries against the databases you want to tune in a format suitable for replay.

For Tuning Advisor to be most effective, a workload should be as realistic as possible. You can generate a workload file or table by performing a trace against your DB Instance. While a trace is running, you can either simulate a load on your DB Instance or run your applications with a normal load.

There are two types of traces: client-side and server-side. A client-side trace is easier to set up and you can watch trace events being captured in real-time in SQL Server Profiler. A server-side trace is more complex to set up and requires some Transact-SQL scripting. In addition, because the trace is written to a file on the RDS DB Instance, storage space is consumed by the trace. It is important to track of how much storage space a running server-side trace uses because the DB Instance could enter a storage-full state and would no longer be available if it runs out of storage space.

For a client-side trace, when a sufficient amount of trace data has been captured in the SQL Server Profiler, you can then generate the workload file by saving the trace to either a file on your local computer or in a database table on an DB Instance that is available to your client computer. The main disadvantage of using a client-side trace is that the trace may not capture all queries when under heavy loads. This could weaken the effectiveness of the analysis performed by the Database Engine Tuning Advisor. If you need to run a trace under heavy loads and you want to ensure that it captures every query during a trace session, you should use a server-side trace.

For a server-side trace, you must get the trace files on the DB Instance into a suitable workload file or you can save the trace to a table on the DB Instance after the trace completes. You can use the SQL Server Profiler to save the trace to a file on your local computer or have the Tuning Advisor read from the trace table on the DB Instance.

Running a Client-Side Trace on a SQL Server DB Instance

To run a client-side trace on a SQL Server DB instance

1. Start SQL Server Profiler. It is installed in the Performance Tools folder of your SQL Server instance folder. You must load or define a trace definition template to start a client-side trace.
2. In the SQL Server Profiler File menu, click **New Trace**. In the **Connect to Server** dialog box, enter the DB Instance endpoint, port, master user name, and password of the database you would like to run a trace on.
3. In the **Trace Properties** dialog box, enter a trace name and choose a trace definition template. A default template, TSQL_Replay, ships with the application. You can edit this template to define your

trace. Edit events and event information under the **Events Selection** tab of the **Trace Properties** dialog box. For more information about trace definition templates and using the SQL Server Profiler to specify a client-side trace see the documentation in MSDN.

4. Start the client-side trace and watch SQL queries in real-time as they execute against your DB Instance.
5. Select **Stop Trace** from the File menu when you have completed the trace. Save the results as a file or as a trace table on you DB Instance.

Running a Server-Side Trace on a SQL Server DB Instance

Writing scripts to create a server-side trace can be complex and is beyond the scope of this document. This section contains sample scripts that you can use as examples. As with a client-side trace, the goal is to create a workload file or trace table that you can open using the Database Engine Tuning Advisor.

The following is an abridged example script that starts a server-side trace and captures details to a workload file. The trace initially saves to the file RDSTrace.trc in the D:\RDSDBDATA\Log directory and rolls-over every 100 MB so subsequent trace files are named RDSTrace_1.trc, RDSTrace_2.trc, etc.

```
DECLARE @file_name NVARCHAR(245) = 'D:\RDSDBDATA\Log\RDSTrace';
DECLARE @max_file_size BIGINT = 100;
DECLARE @on BIT = 1
DECLARE @rc INT
DECLARE @traceid INT

EXEC @rc = sp_trace_create @traceid OUTPUT, 2, @file_name, @max_file_size
IF (@rc != 0) BEGIN
   EXEC sp_trace_setevent @traceid, 10, 1, @on
   EXEC sp_trace_setevent @traceid, 10, 2, @on
   EXEC sp_trace_setevent @traceid, 10, 3, @on
  …
   EXEC sp_trace_setfilter @traceid, 10, 0, 7, N'SQL Profiler'
   EXEC sp_trace_setstatus @traceid, 1
   END
```

The following example is a script that stops a trace. Note that a trace created by the previous script continues to run until you explicitly stop the trace or the process runs out of disk space.

```
DECLARE @traceid INT
SELECT @traceid = traceid FROM ::fn_trace_getinfo(default)
WHERE property = 5 AND value = 1 AND traceid <> 1

IF @traceid IS NOT NULL BEGIN
   EXEC sp_trace_setstatus @traceid, 0
   EXEC sp_trace_setstatus @traceid, 2
END
```

You can save server-side trace results to a database table and use the database table as the workload for the Tuning Advisor by using the fn_trace_gettable function. The following commands load the results of all files named RDSTrace.trc in the D:\rdsdbdata\Log directory, including all rollover files like RDSTrace_1.trc, into a table named RDSTrace in the current database:

```
SELECT * INTO RDSTrace
FROM fn_trace_gettable('D:\rdsdbdata\Log\RDSTrace.trc', default);
```

To save a specific rollover file to a table, for example the RDSTrace_1.trc file, specify the name of the rollover file and substitute 1 instead of default as the last parameter to fn_trace_gettable.

```
SELECT * INTO RDSTrace_1
FROM fn_trace_gettable('D:\rdsdbdata\Log\RDSTrace_1.trc', 1);
```

Running Tuning Advisor with a Trace

Once you create a trace, either as a local file or as a database table, you can then run Tuning Advisor against your RDS instance. Microsoft includes documentation on using the Database Engine Tuning Advisor in MSDN. Using Tuning Advisor with RDS is the same process as when working with a standalone, remote SQL Server instance. You can either use the Tuning Advisor UI on your client machine or use the dta.exe utility from the command line. In both cases, you must connect to the RDS DB Instance using the endpoint for the DB Instance and provide your master user name and master user password when using Tuning Advisor.

The following code example demonstrates using the dta.exe command line utility against an RDS DB Instance with an endpoint of **`dta.cnazcmklsdei.us-east-1.rds.amazonaws.com`**. The example includes the master user name **`admin`** and the master user password **`test`**, the example database to tune is named **`RDSDTA`** and the input workload is a trace file on the local machine named **`C:\RDSTrace.trc`**. The example command line code also specifies a trace session named **`RDSTrace1`** and specifies output files to the local machine named **`RDSTrace.sql`** for the SQL output script, **`RDSTrace.txt`** for a result file, and **`RDSTrace.xml`** for an XML file of the analysis. There is also an error table specified on the RDSDTA database named **`RDSTraceErrors`**.

```
               dta -S dta.cnazcmklsdei.us-east-1.rds.amazonaws.com -U admin -
P test -D RDSDTA -if C:\RDSTrace.trc -s RDSTrace1 -of C:\ RDSTrace.sql -or C:\
 RDSTrace.txt -ox C:\ RDSTrace.xml -e RDSDTA.dbo.RDSTraceErrors
```

Here is the same example command line code except the input workload is a table on the remote RDS instance named **`RDSTrace`** which is on the **`RDSDTA`** database.

```
dta -S dta.cnazcmklsdei.us-east-1.rds.amazonaws.com -U admin -P test -D RDSDTA
 -it RDSDTA.dbo.RDSTrace -s RDSTrace1 -of C:\ RDSTrace.sql -or C:\ RDSTrace.txt
 -ox C:\ RDSTrace.xml -e RDSDTA.dbo.RDSTraceErrors
```

A full list of dta utility command-line parameters can be found in MSDN.

Using SQL Server Agent

With Amazon RDS, you can use SQL Server Agent on a DB Instance running SQL Server Standard, Web Edition, or Enterprise Edition. SQL Server Agent is a Microsoft Windows service that executes

scheduled administrative tasks, which are called jobs. You can use SQL Server Agent to run T-SQL jobs to rebuild indexes, run corruption checks, and aggregate data in a SQL Server DB Instance.

SQL Server Agent can run a job on a schedule, in response to a specific event, or on demand. For more information, see SQL Server Agent in the SQL Server documentation. You should avoid scheduling jobs to run during the maintenance and backup windows for your DB Instance because these maintenance and backup processes that are launched by AWS could interrupt the job or cause it to be cancelled. Because Amazon RDS backs up your DB Instance, you do not use SQL Server Agent to create backups.

To view the history of an individual SQL Server Agent job in the SQL Server Management Studio, you open Object Explorer, right-click the job, and then click **View History**.

Because SQL Server Agent is running on a managed host in a DB Instance, there are some actions that are not supported. Running replication jobs and running command-line scripts by using ActiveX, Windows command shell, or Windows PowerShell are not supported. In addition, you cannot manually start, stop, or restart SQL Server Agent because its operation is managed by the host. Email notifications through SQL Server Agent are not available from a DB Instance.

When you create a SQL Server DB Instance, the master user name is enrolled in the SQLAgentUserRole role. To allow an additional login/user to use SQL Server Agent, you must log in as the master user and do the following.

1. Create another server-level login by using the **CREATE LOGIN** command.
2. Create a user in msdb using **CREATE USER** command, and then link this user to the login that you created in the previous step.
3. Add the user to the SQLAgentUserRole using the `sp_addrolemember` system stored procedure.

For example, suppose your master user name is **`myawsmaster`** and you want to give access to SQL Server Agent to a user named **`theirname`** with a password **`theirpassword`**. You would log in using the master user name and run the following commands.

```
--Initially set context to master database
USE [master];
GO
--Create a server-level login named theirname with password theirpassword
CREATE LOGIN [theirname] WITH PASSWORD = 'theirpassword';
GO
--Set context to msdb database
USE [msdb];
GO
--Create a database user named theirname and link it to server-level login
theirname
CREATE USER [theirname] FOR LOGIN [theirname];
GO
--Added database user theirname in msdb to SQLAgentUserRole in msdb
EXEC sp_addrolemember [SQLAgentUserRole], [theirname];
```

You cannot use the UI in SQL Server Management Console to delete a SQL Server Agent job. To delete a SQL Server Agent job, run the following T-SQL statement.

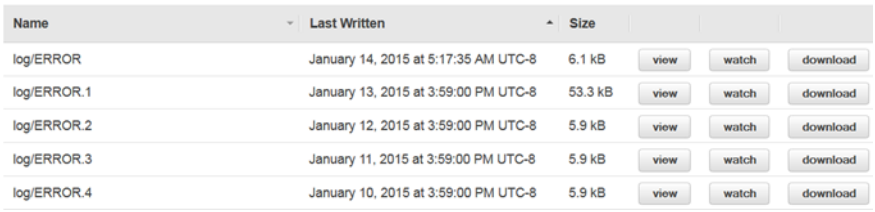
```
EXEC msdb..sp_delete_job @job_name = '<job-name>';
```

Working with SQL Server Logs

You can use the Amazon RDS console to view, watch, and download SQL Server Agent logs and SQL Server error logs.

If you view a log in the Amazon RDS console, you can see its contents as they exist at that moment. Watching a log in the console opens it in a dynamic state so that you can see updates to it in near real time.

Only the latest log will be active for watching. For example, suppose you have the logs shown following:



Name	Last Written	Size			
log/ERROR	January 14, 2015 at 5:17:35 AM UTC-8	6.1 kB	view	watch	download
log/ERROR.1	January 13, 2015 at 3:59:00 PM UTC-8	53.3 kB	view	watch	download
log/ERROR.2	January 12, 2015 at 3:59:00 PM UTC-8	5.9 kB	view	watch	download
log/ERROR.3	January 11, 2015 at 3:59:00 PM UTC-8	5.9 kB	view	watch	download
log/ERROR.4	January 10, 2015 at 3:59:00 PM UTC-8	5.9 kB	view	watch	download

Only log/ERROR, as the most recent log, is being actively updated. You can choose to watch others, but they are static and will not update.

The Amazon RDS console shows logs for the past week through the current day. You can download and archive logs to keep them for reference past that time. One way to archive logs is to load them into an Amazon S3 instance. For instructions on how to set up an Amazon S3 instance and upload a file, go to Amazon S3 Basics in the *Amazon Simple Storage Service Getting Started Guide* and click **Get Started**.

You can also view logs by using the `rdsadmin.dbo.rds_read_error_log` stored procedure. This stored procedure takes two parameters:

- **`@index`**

  Identifies which version of the log to view. Specify 0 to view the current log or 1 to view the previously rotated log.
- **`@type`**

  Identifies which log to view. Specify 1 to view the SQL Server error log or 2 to view the SQL Server Agent log.

For example, to view the current SQL Server Agent log, execute the following statement:

```
EXEC rdsadmin.dbo.rds_read_error_log @index = 0, @type = 2;
```

For more details on viewing, watching, and downloading a log, see the following:

- For details on viewing a log, see Viewing and Listing Database Log Files (p. 605).
- For details on watching a log, see Watching a Database Log File (p. 611).
- For details on downloading a log, see Downloading a Database Log File (p. 609).

Handling UTC Times for Time Zone Awareness

System time is maintained in Universal Coordinated Time (UTC). Amazon RDS does not support changing the time zone for Amazon RDS SQL Server DB instances.

If you want your DB instance to maintain time zone awareness, we recommend using the SQL Server data type `datetimeoffset` to store date and time data. This data type stores the data in UTC time but maintains an offset to identify the local time zone. For example, when using `datetime` the value January 14, 2015, 11:40 AM Pacific Standard Time is saved as `2015-01-14 19:42:18.5770000 -08:00`. Your client application code must be able to convert local time to this format before saving data to the database. For example, in C# you can use the class TimeZoneInfo and the structure DateTimeOffset to get the offset for the local time, then convert local time to UTC time plus an offset.

To convert a `datetimeoffset` value to a local time zone value for display, you can use the SQL Server function CAST in conjunction with the SQL Server functions DATEPART and DATEADD. For example, suppose you have the following table with a `datetimeoffset` field:

```
sales_id    sales_amount    sales_date
1      14.38     2015-01-14 19:42:18.5770000 -08:00
2      39.02     2015-01-14 20:13:51.4330000 -08:00
3      52.66     2015-01-14 20:45:12.1010000 -08:00
```

In this case, you can convert `sales_date` to the local `datetime` value as follows:

```
select sales_id,
sales_amount,
cast((dateadd(mi, datepart(tz, sales_date), sales_date)) AS datetime)
as local_datetime
from sales;
```

The result will look like the following:

```
sales_id sales_amount local_datetime
-------------------------------------------
1   14.38  2015-01-14 11:42:18.577
2   39.02  2015-01-14 12:13:51.433
3   52.66  2015-01-14 12:45:12.100
```

For more information, go to Date and Time Functions in the Microsoft SQL Server documentation.

Appendix: Options for SQL Server Database Engine

This appendix describes options, or additional features, that are available for Amazon RDS instances running the Microsoft SQL Server DB engine. To enable these options, you can add them to an option group, and then associate the option group with your DB instance. For more information about working with options, see Option Groups Overview (p. 510).

The following option is currently supported for SQL Server DB instances:

- SQL Server Transparent Data Encryption (p. 350)
- Multi-AZ Deployment for SQL Server Using the Mirroring Option (p. 353)

SQL Server Transparent Data Encryption

Amazon RDS supports using Transparent Data Encryption (TDE) to encrypt stored data for SQL Server 2008 R2 Enterprise Edition and SQL Server 2012 Enterprise Edition. TDE automatically encrypts data before it is written to storage and automatically decrypts data when the data is read from storage. To enable transparent data encryption for a DB instance that is running SQL Server, specify the **TDE** option in an Amazon RDS option group that is associated with that DB instance.

Transparent data encryption for SQL Server provides encryption key management by using a two-tier key architecture. A certificate, which is generated from the database master key, is used to protect the data encryption keys. The database encryption key performs the actual encryption and decryption of data on the user database. Amazon RDS backs up and manages the database master key and the TDE certificate. To comply with several security standards, Amazon RDS is working to implement automatic periodic master key rotation.

Transparent data encryption is used in scenarios where you need to encrypt sensitive data in case data files and backups are obtained by a third party or when you need to address security-related regulatory compliance issues. Note that you cannot encrypt the system databases for SQL Server, such as the Model or Master databases.

A detailed discussion of transparent data encryption is beyond the scope of this guide, but you should understand the security strengths and weaknesses of each encryption algorithm and key. For information about transparent data encryption for SQL Server, see Transparent Data Encryption (TDE) on the Microsoft website.

You should determine if your DB instance is already associated with an option group that has the **TDE** option. To view the option group that a DB instance is associated with, you can use the RDS console, the rds-describe-db-instance CLI command, or the API action DescribeDBInstances.

The process for enabling transparent data encryption on a SQL Server DB instance is as follows:

1. If the DB instance is not associated with an option group that has the **TDE** option enabled, you must either create an option group and add the **TDE** option or modify the associated option group to add the **TDE** option. For information about creating or modifying an option group, see Working with Option Groups (p. 510). For information about adding an option to an option group, see Adding an Option to an Option Group (p. 515).
2. Associate the DB instance with the option group with the **TDE** option. For information about associating a DB instance with an option group, see Modifying a DB Instance Running the SQL Server Database Engine (p. 328).

When the **TDE** option is added to an option group, Amazon RDS generates a certificate that is used in the encryption process. You can then use the certificate to run SQL statements that will encrypt data in a database on the DB instance. The following example uses the RDS-created certificate called `RDSTDECertificateName` to encrypt a database called `customerDatabase`.

```
---------- Enabling TDE -------------

-- Find a RDSTDECertificate to use
USE [master]
GO
SELECT name FROM sys.certificates WHERE name LIKE 'RDSTDECertificate%'
GO

USE [customerDatabase]
GO
-- Create DEK using one of the certificates from the previous step
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE [RDSTDECertificateName]
GO

-- Enable encryption on the database
ALTER DATABASE [customerDatabase]
SET ENCRYPTION ON
GO

-- Verify that the database is encrypted
USE [master]
GO
SELECT name FROM sys.databases WHERE is_encrypted = 1
GO
SELECT db_name(database_id) as DatabaseName, * FROM sys.dm_database_encryp
tion_keys
GO
```

The time it takes to encrypt a SQL Server database using TDE depends on several factors, including the size of the DB instance, whether PIOPS is enabled for the instance, the amount of data, and other factors.

The **TDE** option is a persistent option than cannot be removed from an option group unless all DB instances and backups are disassociated from the option group. Once you add the **TDE** option to an option group, the option group can only be associated with DB instances that use TDE. For more information about persistent options in an option group, see Option Groups Overview (p. 510).

Because the **TDE** option is a persistent option, you can also inadvertently have a conflict between the option group and an associated DB instance. You can have a conflict between the option group and an associated DB instance in the following situations:

- The current option group has the **TDE** option, and you replace it with an option group that does not have the **TDE** option.
- You restore a DB instance that no longer uses TDE from a point-in-time DB snapshot that was taken when the DB instance was using TDE. The option group for the DB instance that no longer uses TDE will conflict with the restored DB instance that uses TDE.

To disable TDE for a DB instance, first ensure that there are no encrypted objects left on the DB instance by either unencrypting the objects or by dropping them. If any encrypted objects exist on the DB instance, you will not be allowed to disable TDE for the DB instance. When using the RDS Console to remove the **TDE** option from an option group, the console will indicate it is processing and an event will be created indicating an error if the option group is associated with an encrypted DB instance or DB snapshot.

The following example removes the TDE encryption from a database called `customerDatabase`.

```
------------- Removing TDE ----------------

USE [customerDatabase]
GO

-- Disable encryption on the database
ALTER DATABASE [customerDatabase]
SET ENCRYPTION OFF
GO

-- Wait until the encryption state of the database becomes 1. The state will
be 5 (Decryption in progress) for a while
SELECT db_name(database_id) as DatabaseName, * FROM sys.dm_database_encryp
tion_keys
GO

-- Drop the DEK used for encryption
DROP DATABASE ENCRYPTION KEY
GO

-- Alter to SIMPLE Recovery mode so that your encrypted log gets truncated
USE [master]
GO
ALTER DATABASE [customerDatabase] SET RECOVERY SIMPLE
GO
```

When all objects are unencrypted, you can modify the DB instance to be associated with an option group without the **TDE** option or you can remove the **TDE** option from the option group.

Performance Considerations

The performance of a SQL Server DB instance can be impacted by using transparent data encryption.

Performance for unencrypted databases can also be degraded if the databases are on a DB instance that has at least one encrypted database. As a result, we recommend that you keep encrypted and unencrypted databases on separate DB instances.

Because of the nature of encryption, the database size and the size of the transaction log will be larger than for an unencrypted database. You could run over your allocation of free backup space. The nature of TDE will cause an unavoidable performance hit. If you need high performance and TDE, measure the impact and make sure it meets your needs. There is less of an impact on performance if you use Provisioned IOPS and at least an M3.Large DB instance class.

Multi-AZ Deployment for SQL Server Using the Mirroring Option

Amazon RDS supports Multi-AZ deployments for SQL Server using the Mirroring option. In a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. For more information about Multi-AZ deployments, see High Availability (Multi-AZ) (p. 71). For an overview of SQL Server Mirroring and Amazon RDS, see Planning your Multi-AZ Deployments Using SQL Server Mirroring (p. 307)

> **Note**
> SQL Server Multi-AZ using Mirroring is currently available in the US East (N. Virginia), US West (Oregon), and EU (Ireland) AWS regions. We plan to support other regions in the future.

The process for enabling Multi-AZ using Mirroring as an option for a SQL Server DB instance is as follows:

1. If your DB instance is not associated with an option group that has the Mirroring option enabled, you can do one of three tasks:
   - Create an option group and add the **Mirroring** option. For information about creating an option group, see Working with Option Groups (p. 510).
   - Modify the option group currently associated with the DB instance to add the **Mirroring** option. For information about adding an option to an option group, see Adding an Option to an Option Group (p. 515).
   - Associate one of the default option groups that have the **Mirroring** option already added. These option groups are available for each engine version and edition combination, such as **default:sqlserver-se-10-50-mirrored** or **default:sqlserver-se-11-00-mirrored**.
2. Associate the DB instance with the option group with the **Mirroring** option. For information about associating a DB instance with an option group, see Modifying a DB Instance Running the SQL Server Database Engine (p. 328).

We recommend that you create an option group with the **Mirroring** option and then associate the option group with the SQL Server DB instances you want to use with Multi-AZ with Mirroring. The following RDS CLI examples create a SQL Server option group, then adds the **Mirroring** option to the option group, and then associates that option group with a SQL Server DB instance.

The following RDS CLI example creates an option group named *MirroringOG* for SQL Server SE 10.50:

```
PROMPT> rds-create-option-group MirroringOG --engine-name sqlserver-se --major-
engine-version 10.50 --description "SQLServer Mirroring"
```

The following RDS CLI example adds the **Mirroring** option to an option group named *MirroringOG*:

```
PROMPT> rds-add-option-to-option-group MirroringOG --option-name Mirroring
```

You can then associate an SQL Server DB instance with the option group. The following RDS CLI example associates a SQL Server DB instance named *cust_instance_id* with an option group named *MirroringOG*:

```
PROMPT> rds-modify-db-instance cust_instance_id -og MirroringOG --apply-immedi
```

```
ately
```

When the **Mirroring** option is added to an option group, Amazon RDS begins the replication and synchronization process for any SQL Server DB instances that are associated with the option group.

Removing Multi-AZ (Mirroring) from a SQL Server DB Instance

SQL Server Mirroring is enabled in Amazon RDS through the use of option groups. When you create a SQL Server DB instance that uses Multi-AZ with Mirroring, Amazon RDS automatically creates a default option group that has the **Mirroring** option enabled. If you need to disable mirroring to perform a bulk load or other tasks that do not require mirroring, do the steps below. Note that you can't just remove the **Mirroring** option from the option group associated with the DB instance.

To disable mirroring for a SQL Server DB instance, do the following:

1. Create a new option group that doesn't have the **Mirroring** option.
2. Associate this new option group with the SQL Server DB instance using mirroring. This will cause the instance to stop mirroring and become a single Availability Zone instance.

PostgreSQL on Amazon RDS

Amazon RDS supports DB instances running several versions of PostgreSQL. You can create DB instances and DB snapshots, point-in-time restores and backups. DB instances running PostgreSQL support Multi-AZ deployments, Read Replicas (version 9.3.5 and later), Provisioned IOPS, and can be created inside a VPC. You can also use SSL to connect to a DB instance running PostgreSQL.

Before creating a DB instance, you should complete the steps in the Setting Up for Amazon RDS (p. 7) section of this guide.

You can use any standard SQL client application to run commands for the instance from your client computer. Such applications include *pgAdmin*, a popular Open Source administration and development tool for PostgreSQL, or *psql*, a command line utility that is part of a PostgreSQL installation. In order to deliver a managed service experience, Amazon RDS does not provide host access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application. Amazon RDS does not allow direct host access to a DB instance via Telnet or Secure Shell (SSH).

These are the common management tasks you perform with a PostgreSQL DB instance, with links to information about each task:

- For planning information, such as PostgreSQL versions, storage engines, security, and features supported in Amazon RDS, see Amazon RDS PostgreSQL Planning Information (p. 356).
- There are prerequisites you must complete before you create your DB instance; for more information, see the Setting Up for Amazon RDS (p. 7) section of this guide. For example, DB instances are created by default with a firewall that prevents access to it. You therefore must create a security group with the correct IP addresses and network configuration you will use to access the DB instance.
- If you are creating a DB instance for production purposes, you should understand how instance classes, storage, and Provisioned IOPS work in Amazon RDS. For more information about DB instance classes, see DB Instance Class (p. 65) For more information about Amazon RDS storage, see Amazon RDS Storage Types (p. 77). For more information about Provisioned IOPS, see Amazon RDS Provisioned IOPS Storage to Improve Performance (p. 82).
- A production DB instance should also use Multi-AZ deployments. All Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances. For more information about Multi-AZ deployments, see High Availability (Multi-AZ) (p. 71).
- You can create a PostgreSQL Read Replica (master/standby) DB instance to service read-only traffic. For more information about PostgreSQL Read Replicas, see Working with PostgreSQL and MySQL Read Replicas (p. 472).
- If your AWS account has a default VPC (a default virtual private network), then your DB instance will automatically be created inside the default VPC. If your account does not have a default VPC and you

want the DB instance to be inside a VPC, you must create the VPC and subnet groups before you create the DB instance. For more information about determining if your account has a default VPC, see Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 563). For more information about using VPCs with Amazon RDS, see Using Amazon RDS with Amazon Virtual Private Cloud (VPC) (p. 562).

- If you want to modify PostgreSQL database parameters, you must create a parameter group and assign it to your DB instance. For more information on parameter groups, see Working with DB Parameter Groups (p. 523).
- After you create a security group and associate it to a DB instance, you can connect to the DB instance using any standard SQL client application such as *pgAdmin*. For more information on connecting to a DB instance, see Connecting to a DB Instance Running the PostgreSQL Database Engine (p. 370).
- You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots. For information, see Backing Up and Restoring (p. 495).
- You can monitor an instance through actions such as viewing the PostgreSQL logs, CloudWatch Amazon RDS metrics, and events. For information, see Monitoring Amazon RDS (p. 571).

There is also an important appendix with useful information about working with PostgreSQL DB instances. For information on common DBA tasks for PostgreSQL on Amazon RDS, see Appendix: Common DBA Tasks for PostgreSQL (p. 380).

Amazon RDS PostgreSQL Planning Information

Amazon RDS supports DB instances running several editions of PostgreSQL. This section shows how you can work with PostgreSQL on Amazon RDS. You should also be aware of the limits for PostgreSQL DB instances.

For information about importing PostgreSQL data into a DB instance, see Importing Data into PostgreSQL on Amazon RDS (p. 377).

Topics

- Supported PostgreSQL Database Versions (p. 357)
- Database Engine Features (p. 359)
- Limits for PostgreSQL DB Instances (p. 361)
- Minor Version Upgrades (p. 361)
- Using SSL with a PostgreSQL DB Instance (p. 361)

When you create a DB instance, the master user system account that you create is assigned to the `rds_superuser` role. The `rds_superuser` role is similar to the PostgreSQL superuser role (customarily named postgres in local instances) but with some restrictions. As with the PostgreSQL superuser role, the `rds_superuser` role has the most privileges on your DB instance and you should not assign this role to users unless they need the most access to the DB instance.

The `rds_superuser` role can do the following:

- Add extensions that are available for use with Amazon RDS
- Manage tablespaces, including creating and deleting them
- View all users not assigned the `rds_superuser` role using the `pg_stat_activity` command and kill their connections using the `pg_terminate_backend` and `pg_cancel_backend` commands.
- Grant and revoke the replication attribute onto all roles that are not the `rds_superuser` role

Tablespaces are supported in PostgreSQL on Amazon RDS for compatibility; since all storage is on a single logical volume, tablespaces cannot be used for IO splitting or isolation. We have benchmarks and practical experience that shows that a single logical volume is the best setup for most use cases.

The PostgreSQL auto-vacuum is an optional, but highly recommended, parameter that by default is turned on for new PostgreSQL DB instances. Do not turn this parameter off. For more information on using auto-vacuum with Amazon RDS PostgreSQL, see Best Practices for Working with PostgreSQL (p. 60).

To import PostgreSQL data into a DB instance, follow the information in the Importing Data into PostgreSQL on Amazon RDS (p. 377) section.

Supported PostgreSQL Database Versions

Currently Amazon RDS supports PostgreSQL version 9.4.4 as well as versions 9.4.1, 9.3.1, 9.3.2, 9.3.3, 9.3.5, 9.3.6, and 9.3.9.

PostgreSQL Version 9.4.4

PostgreSQL version 9.4.4 contains fixes from previous releases as well as fixes to those previous releases. All PostgreSQL update releases are cumulative, and version 9.4.4 fixes a number of problems inadvertently introduced by fixes in earlier versions. We strongly urge users to upgrade to this version, rather than installing less recent versions that have known issues. Version 9.4.4 closes multiple known bugs with multixact handling, and the PostgreSQL Project does not anticipate additional update releases soon. For more information about PostgreSQL version 9.4.4, see PostgreSQL 9.4.4, 9.3.9, 9.2.13, 9.1.18 and 9.0.22 Released!.

This release includes updates from previous versions, including the following:

- Security fixes added in version 9.4.2. For more information about fixes in version 9.4.2, see PostgreSQL 9.4.2, 9.3.7, 9.2.11, 9.1.16, and 9.0.20 released!.
- Data corruption fixes for "multixact wraparound" added in version 9.4.2 (that were subsequently fixed in version 9.4.4).
- File permissions fix added in version 9.4.3. For more information about fixes in version 9.4.3, see PostgreSQL 9.4.3, 9.3.8, 9.2.12, 9.1.17 and 9.0.21 Released!.

PostgreSQL Version 9.4.1

PostgreSQL version 9.4.1 contains multiple security updates, including several patches to buffer overruns. Version 9.4.1 also includes a change in the way Unicode strings are escaped for the JSON and JSONB data types. For more information on the 9.4.1 release, see the PostgreSQL documentation and the PostgreSQL wiki.

The new PostgreSQL versions for Amazon RDS also includes the following:

- JSONB data type - The ability to include JSON-formatted fields in PostgreSQL tables give you more flexibility when managing schemas. JSONB items are stored in a decomposed binary format that speeds query operations. For more information on using the JSONB data type with a PostgreSQL database, see the PostgreSQL documentation.
- pg_prewarm - When a database instance is restarted, its shared buffers are empty, which means that all queries will initially have to read data direct from disk. The pg_prewarm module can be used to load relation data back into the buffers to "warm" the buffers back up again. This means that queries that would otherwise have to load parts of a table in bit by bit can have the data available in shared buffers ready for use. For more information on pg_warm, see the PostgreSQL documentation.
- PostGIS version 2.1.5
- plv8 version 1.4.3

In PostgreSQL 9.4.1 on Amazon RDS, the `fsync` and `full_page_writes` database parameters are not modifiable. Disabling the `fsync` and `full_page_writes` database parameters can lead to data corruption, so we have enabled them for you. We recommend that customers with other 9.3 DB engine versions of PostgreSQL not disable the `fsync` and `full_page_writes` parameters.

To create a new PostgreSQL 9.4.1 DB instance, select the DB engine version "9.4.1" when you use the *Launch DB Instance Wizard* in the RDS console. Amazon RDS does not currently support an in-place upgrade from a PostgreSQL 9.3.x DB instance to a PostgreSQL 9.4.1 DB instance. However, you can migrate a database from a PostgreSQL 9.3.5 DB instance to a PostgreSQL 9.4.1 DB instance by using the following steps:

1. Create a new PostgreSQL 9.4.1 DB instance
2. Create a backup of your existing PostgreSQL 9.3.5 database using *pg_dump*
3. Import the dump file into your PostgreSQL 9.4.1 DB instance using *pg_restore*

Be sure to test your application against the new version of PostgreSQL before going into production.

PostgreSQL Version 9.3.9

PostgreSQL version 9.3.9 contains fixes from previous releases as well as fixes to those previous releases. All PostgreSQL update releases are cumulative, and version 9.3.9 fixes a number of problems inadvertently introduced by fixes in earlier versions. We strongly urge users to upgrade to this version, rather than installing less recent versions that have known issues. Version 9.3.9 closes multiple known bugs with multixact handling, and the PostgreSQL Project does not anticipate additional update releases soon. For more information about PostgreSQL version 9.3.9, see PostgreSQL 9.4.4, 9.3.9, 9.2.13, 9.1.18 and 9.0.22 Released!.

This release includes updates from previous versions, including the following:

- Security fixes added in version 9.3.7. For more information about fixes in version 9.3.7, see PostgreSQL 9.4.2, 9.3.7, 9.2.11, 9.1.16, and 9.0.20 released!.
- Data corruption fixes for "multixact wraparound" added in version 9.3.7 (that were subsequently fixed in version 9.3.9).
- File permissions fix added in version 9.3.8. For more information about fixes in version 9.3.8, see PostgreSQL 9.4.3, 9.3.8, 9.2.12, 9.1.17 and 9.0.21 Released!.

PostgreSQL Version 9.3.6

PostgreSQL version 9.3.6 contains multiple security updates, including several patches to buffer overruns.

The new PostgreSQL versions for Amazon RDS also includes the following:

- PostGIS version 2.1.5
- plv8 version 1.4.3

To upgrade your DB instance to version 9.3.6, modify the DB instance using the RDS console, RDS CLI, or RDS API, and select version 9.3.6 as the new DB engine version. To use the latest version of PostGIS and plv8, use the ALTER EXTENSION UPDATE statement to update after you upgrade to version 9.3.6.

PostgreSQL Version 9.3.5

PostgreSQL version 9.3.5 has several important changes, including the following:

- Adds support for Read Replicas. For more information on PostgreSQL Read Replicas, see Working with PostgreSQL and MySQL Read Replicas (p. 472)
- Allows the `rds_superuser` role to set the `session_replication_role` parameter. This change means that you can use open source, trigger-based replication tools such as Londiste to migrate existing PostgreSQL data to Amazon RDS with minimal downtime.

  You can also use the `session_replication_role` parameter to run a replica of your PostgreSQL DB instance on an on-premises server or on an EC2 instance. For example, you could install Bucardo, an open source trigger-based lazy replication solution, on a remote instance and set it as a replica to a source PostgreSQL DB instance.

  For more information about using the `session_replication_role` parameter, see this blog post.
- Adds the PostGIS version 2.1.3 extension.
- Expands access to `pg_stat_statements`. Users can view the performance of the queries they execute. Users granted the `rds_superuser` privileges can view all user queries and can reset all queries tracked by `pg_stat_statements`.

  You can view `pg_stat_statements` by setting the SHARED_PRELOAD_LIBRARIES parameter to **`pg_stat_statements`**. In previous PostgreSQL versions on Amazon RDS, changing this setting was not allowed.

  The `rds_superuser` role includes privileges for the following commands:
  - pg_stat_reset
  - pg_stat_statements
  - pg_stat_statements_reset
  - pg_stat_replication

  Important
  If you executed the `CREATE EXTENSION pg_stat_statements;` statement on your RDS Postgres instance when it was running version 9.3.3, you will need to drop and recreate the extension when you upgrade to version 9.3.5. The create extension command on version 9.3.5 will grant the correct privileges to `rds_superuser`.

```
DROP EXTENSION pg_stat_statements;
CREATE EXTENSION pg_stat_statements;
```

- Adds support for the *PL/V8* extension, which is a PostgreSQL procedural language extension that lets you write JavaScript functions that can be called from SQL.
- Adds support for the *postgres_fdw* extension, which gives you the ability to access and modify data stored on other PostgreSQL servers as if the data was in tables on your Amazon RDS PostgreSQL DB instance.

Database Engine Features

PostgreSQL uses extensions that allow related pieces of functionality, such as datatypes and functions, to be bundled together and installed in a database with a single command. Note that the XML data type is currently supported only in version 9.3.2 and later.

The following list shows a subset of the key PostgreSQL extensions that are currently supported by PostgreSQL on Amazon RDS. For more information on PostgreSQL extensions, see Packaging Related Objects into an Extension.

- Data Type Extensions:

  - hstore - Provides a key/value pair store.
  - citext - Provides a case-insensitive character string type.
  - ltree - Provides a data type for representing labels of data stored in a hierarchical tree-like structure.
  - isn - Provides data types for international product numbering standards such as EAN13, UPC, ISSN, and ISBN.
  - cube - Provides a data type for representing multidimensional cubes.
- Full Text Search Dictionaries:
  - dict_int - An add-on dictionary template for full-text search often used to control the indexing of integers.
  - unaccent - A text search dictionary that removes accents (diacritic signs) from lexemes.
- PostGIS, postgis_tiger_geocoder, and postgis_topology - Spatial and geographic objects for PostgreSQL.
- dblink- Supports connections to other PostgreSQL databases from within a database session.
- Misc Extensions
  - earthdistance - Calculates great circle distances on the surface of the Earth.
  - fuzzystrmatch - Determines similarities and distance between strings.
  - intarray - Provides functions and operators for manipulating null-free arrays of integers.
  - postgres_fdw - (Version 9.3.5 or later) Foreign-data wrapper that can be used to access data stored on external PostgreSQL servers.
  - pg_stat_statements - (Version 9.3.5 or later) Provides a means for tracking execution statistics of all SQL statements executed.
  - pgcrypto- Provides cryptographic functions.
  - pg_trgm - Functions that determine the similarity of alphanumeric text based on trigram matching.
  - tablefunc - Provides various functions that return tables.
  - uuid-ossp - Generates UUIDs (does requires the OSSP UUID library, which can be found at http://www.ossp.org/pkg/lib/uuid/ - MIT License).
  - btree_gin - Provides a sample GIN operator that uses B-tree-like behavior for certain data types.
  - chkpass - Provides a data type designed for storing encrypted passwords.
  - intagg - Provides an integer aggregator and enumerator. This module is now obsolete but still provides a compatible wrapper around the built-in functions that superseded it.
  - tsearch2 - Provides backwards-compatible text search functionality.
  - pgrowlocks - Provides row locking information for a specified table.
  - sslinfo - Provides information about the SSL certificate provided by the current client when it connected to PostgreSQL.
- Index Types
  - btree_gist - Provides GiST index operator classes that implement B-tree.
- Supported PL languages include:
  - PL/pgSQL
  - PL/Tcl
  - PL/Perl
  - PL/V8 (Version 9.3.5 and later)

The current list of extensions supported by Amazon RDS can be found in the default DB parameter group for PostgreSQL, called "default.postgres9.3." You can see the current extensions list using psql by showing the rds.extensions parameter like in the following example:

```
SHOW rds.extensions;
```

Limits for PostgreSQL DB Instances

You can have up to 40 PostgreSQL DB instances. The following is a list of limitations for PostgreSQL on Amazon RDS:

- The minimum storage size for a PostgreSQL DB instance is 5 GB.
- The maximum storage size for a PostgreSQL DB instance is 6 TB for all instances.
- Amazon RDS reserves up to 3 connections for system maintenance. If you specify a value for the user connections parameter, you will need to add 3 to the number of connections that you expect to use.

Minor Version Upgrades

With Amazon RDS, you can control when to upgrade your PostgreSQL instance to new versions supported by Amazon RDS. You can maintain compatibility with specific PostgreSQL versions, test new versions with your application before deploying in production, and perform version upgrades on your own terms and timelines.

Unless you specify otherwise, your DB Instance will automatically be upgraded to new PostgreSQL minor versions as they are supported by Amazon RDS. This patching will occur during your scheduled maintenance window, and it will be announced on the Amazon RDS Community Forum in advance. To turn off automatic version upgrades, set the `AutoMinorVersionUpgrade` parameter for your DB instance to `false`.

Using SSL with a PostgreSQL DB Instance

Amazon RDS supports SSL encryption for PostgreSQL DB instances. Using SSL, you can encrypt a PostgreSQL connection between your applications and your PostgreSQL DB instances. SSL support is available in all AWS regions for PostgreSQL. Amazon RDS creates an SSL certificate for your PostgreSQL DB instance when the instance is created. If you enable SSL certificate verification, then the SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks.

Important
Amazon RDS will rotate all SSL certificates for DB instances on March 23, 2015 but will not initiate a reboot of the instance. If you use SSL to connect to an Amazon RDS DB instance, you must follow the steps in the topic SSL Certificate Rotation (p. 109) to apply a new SSL certificate to your DB instance before March 23, 2015 or you will not be able to connect to the DB instance using SSL.

To use a PostgreSQL DB instance over SSL, follow these general steps:

1. Download the public key stored at http://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem.
2. Import the certificate into your operating system.
3. Connect to your PostgreSQL DB instance over SSL by appending `sslmode=require` to your connection string. Use the `sslrootcert` parameter to reference the public key, for example, `sslrootcert=rds-ssl-ca-cert.pem`.
4. Instead of `sslmode=require`, use `sslmode=verify-full` to have the SSL connection verify the DB instance endpoint against the endpoint in the SSL certificate.

Note
Prior to August 5, 2014, SSL certificate verification was not available and SSL certificates for PostgreSQL DB instances did not use the DB instance endpoint as the CN for the SSL certificate for the DB instance. If you have a PostgreSQL DB instance that was created before August 5,

2014, and you want to ensure that the instance endpoint is included as the CN for the SSL certificate for that DB instance, then rename the DB instance. When you rename a DB instance, a new certificate is deployed for the DB instance and the instance is rebooted to enable the new certificate.
The SSL certificate verification `sslmode=verify-full` connection string parameter is not valid for connections prior to August 5, 2014.

The encrypted status of your connection is shown when you connect to the DB instance in the logon banner:

```
Password for user master:
psql (9.3.1)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

You can also load the `sslinfo` extension and then call the `ssl_is_used()` function to determine if SSL is being used. The function returns true (t) if the connection is using SSL, otherwise it returns false (f).

```
postgres=> create extension sslinfo;
CREATE EXTENSION

postgres=> select ssl_is_used();
 ssl_is_used
-------------
 t
(1 row)
```

If the SSL parameter is set to true (the default) in the associated parameter group, you can also show the parameter value using the following command:

```
postgres=> show ssl;
 ssl
-----
 on
(1 row)
```

Creating a DB Instance Running the PostgreSQL Database Engine

The basic building block of Amazon RDS is the DB instance. This is the environment in which you will run your PostgreSQL databases.

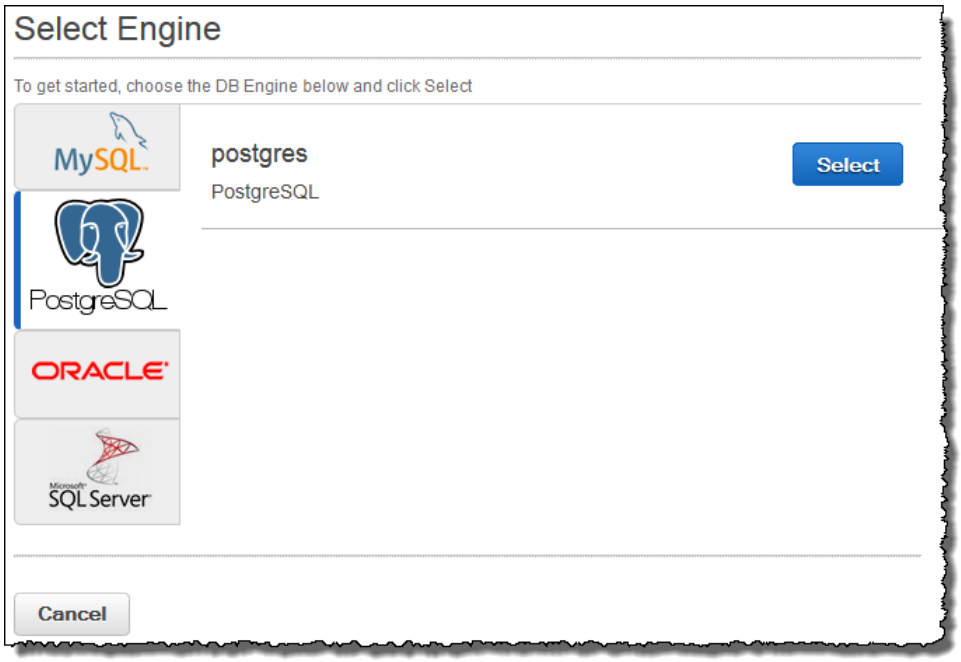
Important
You must complete the tasks in the Setting Up for Amazon RDS (p. 7) section before you can create or connect to a DB instance.

AWS Management Console

To launch a PostgreSQL DB instance

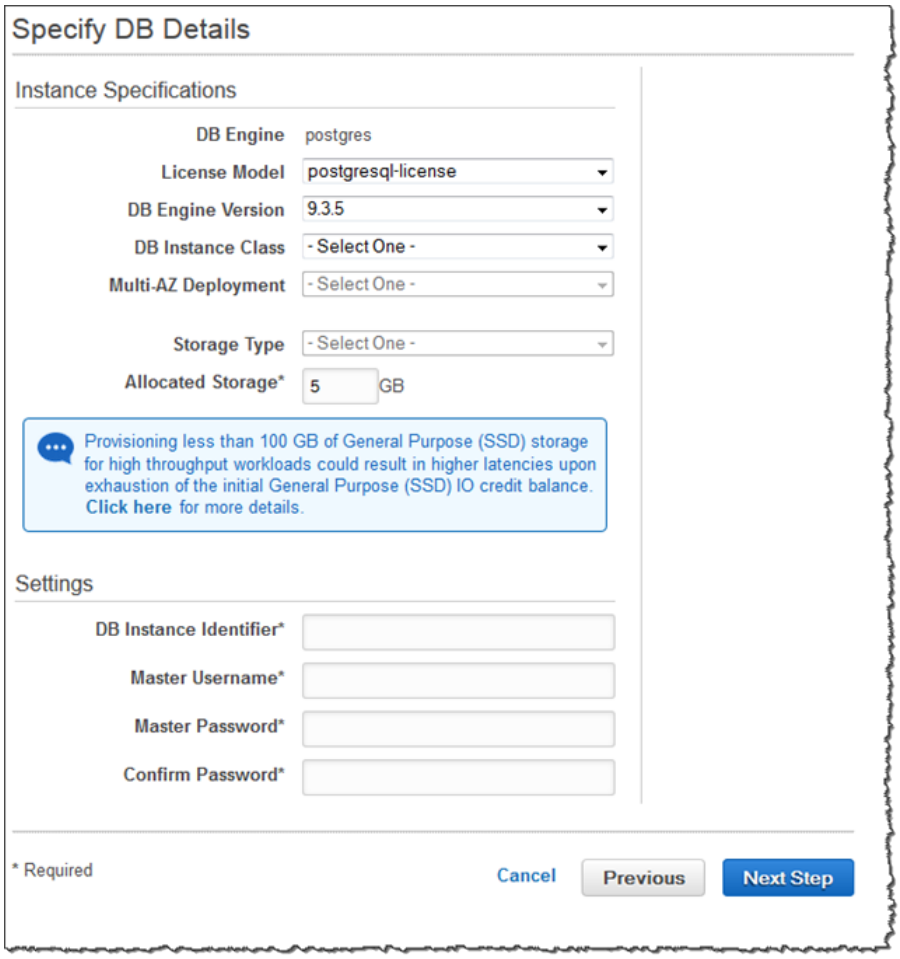
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the top right corner of the AWS Management Console, select the region where you want to create the DB instance.
3. In the navigation pane, click **DB Instances**.
4. Click **Launch DB Instance** to start the **Launch DB Instance Wizard**.

   The wizard opens on the **Select Engine** page.



5. On the **Select Engine** page, click the PostgreSQL icon and then click the **Select** button for the PostgreSQL DB engine.
6. Next, the **Production?** page asks if you are planning to use the DB instance you are creating for production. If you are, select **Yes**. By selecting **Yes**, the failover option **Multi-AZ** and the **Provisioned IOPS** storage option will be preselected in the following step. Click **Next** when you are finished.
7. On the **Specify DB Details** page, specify your DB instance information. Click **Next** when you are finished.

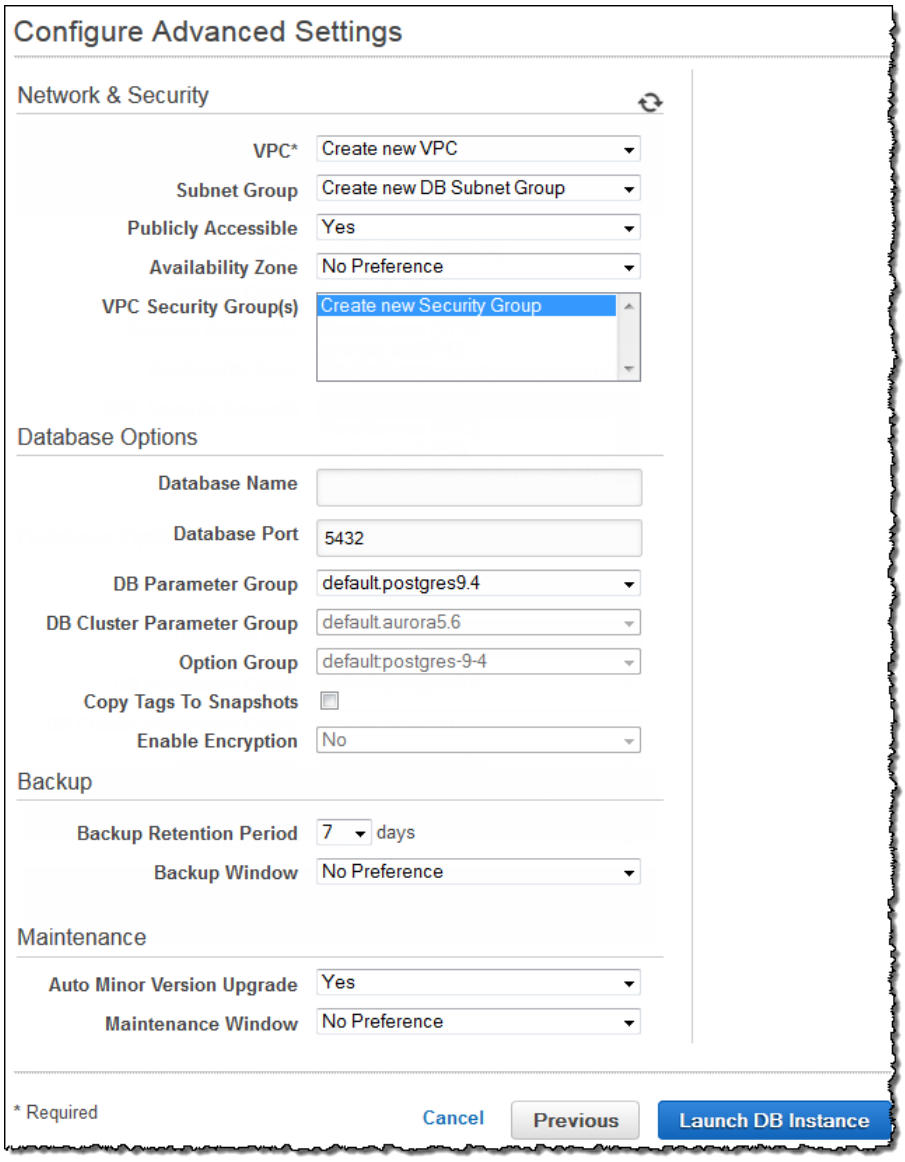
For this parameter...	...Do this:
License Model	PostgreSQL has only one license model. Select the default, **`postgresql-license`**, to use the general license agreement for PostgreSQL.
DB Engine Version	Select the version of PostgreSQL that you want to work with.
DB Instance Class	Select a DB instance class that defines the processing and memory requirements for the DB instance. For more information about all the DB instance class options, see DB Instance Class (p. 65).
Multi-AZ Deployment	Determine if you want to create a standby replica of your DB instance in another Availability Zone for failover support. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 69).
Allocated Storage	Type a value to allocate storage for your database (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. The minimum allocated storage for a PostgreSQL instance is 5 GB. For more information about storage allocation, see Amazon Relational Database Service Features.
Storage Type	Select the storage type you want to use. For more information about storage, see Storage for Amazon RDS (p. 77).
DB Instance Identifier	Type a name for the DB instance that is unique for your account in the region you selected. You may chose to add some intelligence to the name such as including the region and DB engine you selected, for example **`postgresql-instance1`**.
Master Username	Type a name using alphanumeric characters that you will use as the master user name to log on to your DB instance. For information on the default privileges granted to the master user name, see Amazon RDS PostgreSQL Planning Information (p. 356)
Master Password and **Confirm Password**	Type a password that contains from 8 to 128 printable ASCII characters (excluding /,", and @) for your master user password. Retype the password in the **Confirm Password** text box.



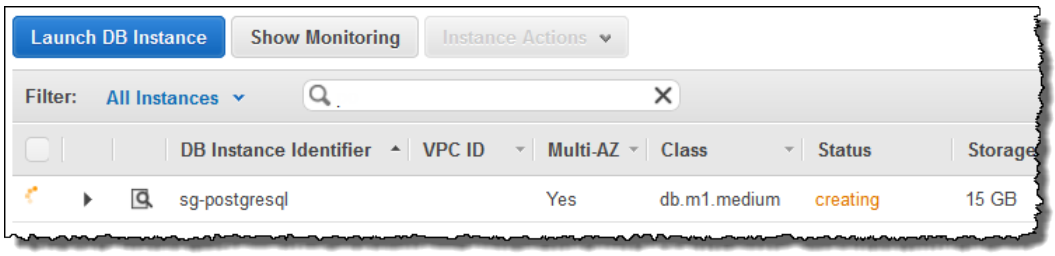
8. On the **Configure Advanced Settings** page, provide additional information that Amazon RDS needs to launch the PostgreSQL DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then click Launch DB Instance.

For this parameter...	...Do this:
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, select the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, select **`Not in VPC`**. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).
DB Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, select **`default`**, which will be the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, select the DB subnet group you created for that VPC. For more information about VPC, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).

For this parameter...	...Do this:
Publicly Accessible	Select **Yes** to give the DB instance a public IP address, meaning that it will be accessible outside the VPC; otherwise, select **No**, so the DB instance will only be accessible from inside the VPC. For more information about hiding DB instances from public access, see Hiding a DB instance in a VPC from the Internet.
Availability Zone	Use the default value of **No Preference** unless you want to specify an Availability Zone.
VPC Security Group	If you are a new customer to AWS, select the default VPC. If you created a VPC security group, select the VPC security group you previously created.
Database Name	If you want to specify a database name for the default database, type a name for your database of up to 63 alphanumeric characters. If you do not provide a name, the default "postgres" database is created.
Database Port	Specify a port you want to use to access the database. PostgreSQL installations default to port 5432 .
Parameter Group	Select a parameter group. Each PostgreSQL version has a default parameter group you can use, or you can create your own parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 523).
Option Group	Option groups are currently not used with PostgreSQL DB instances. For more information about option groups, see Working with Option Groups (p. 510).
Enable Encryption	Select Yes to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 106).
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For non-trivial instances set this value to **1** or greater.
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of **No Preference**.
Auto Minor Version Upgrade	Select **Yes** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Maintenance Window	Select the 30 minute window in which pending modifications to your DB instance are applied. If you the time period doesn't matter, select **No Preference**.



9. On the final page of the wizard, click **Close**.
10. On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new instance to be available.



CLI

To create a PostgreSQL DB instance

- Use the command `rds-create-db-instance` to create a DB instance.

```
PROMPT>rds-create-db-instance pgdbinstance -s 20 -c db.m1.small -e postgresql

   - u <masterawsuser> -p <masteruserpassword>
```

This command should produce output similar to the following:

```
DBINSTANCE  pgdbinstance  db.m1.small  postgresql  20  sa  creating  3  ****
  n  9.3
SECGROUP  default  active
PARAMGRP  default.PostgreSQL9.3  in-sync
```

API

To create a PostgreSQL DB instance

- Call the `CreateDBInstance` action. For example, you could use the following parameters:
  - *`DBEngine`* = `postgresql`
  - *`DBInstanceIdentifier`* = `pgdbinstance`
  - *`DBInstanceClass`* = `db.m1.small`
  - *`AllocatedStorage`* = `20`
  - *`BackupRetentionPeriod`* = `3`
  - *`MasterUsername`* = `<masterawsuser>`
  - *`MasterUserPassword`* = `<masteruserpassword>`

Example

```
https://rds.amazonaws.com/
   ?Action=CreateDBInstance
   &AllocatedStorage=20
   &BackupRetentionPeriod=3
   &DBInstanceClass=db.m1.small
   &DBInstanceIdentifier=pgdbinstance
   &DBName=mydatabase
   &DBSecurityGroups.member.1=mysecuritygroup
   &DBSubnetGroup=mydbsubnetgroup
   &Engine=postgresql
   &MasterUserPassword=<masteruserpassword>
   &MasterUsername=<masterawsuser>
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140212/us-west-2/rds/aws4_request
   &X-Amz-Date=20140212T190137Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=60d520ca0576c191b9eac8dbfe5617ebb6a6a9f3994d96437a102c0c2c80f88d
```

Related Topics

- Amazon RDS DB Instances (p. 64)
- DB Instance Class (p. 65)
- Deleting a DB Instance (p. 459)

Connecting to a DB Instance Running the PostgreSQL Database Engine

After Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. It is important to note that the security group you assigned to the DB instance when you created it must allow access to the DB instance. If you have difficulty connecting to the DB instance, the problem is most often with the access rules you set up in the security group you assigned to the DB instance.

This section shows two ways to connect to a PostgreSQL DB instance. The first example uses *pgAdmin*, a popular Open Source administration and development tool for PostgreSQL. You can download and use *pgAdmin* without having a local instance of PostgreSQL on your client computer. The second example uses *psql*, a command line utility that is part of a PostgreSQL installation. To use *psql*, you must have a PostgreSQL installed on your client computer or have installed the *psql* client on your machine.

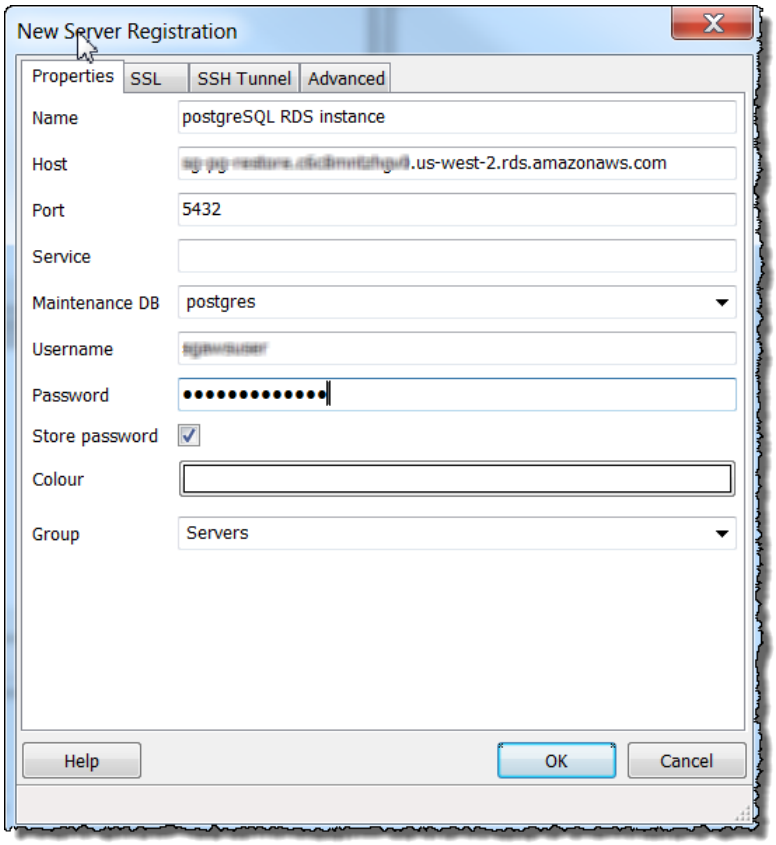
In this example, you connect to a PostgreSQL DB instance using pgAdmin.

Using pgAdmin to Connect to a PostgreSQL DB Instance

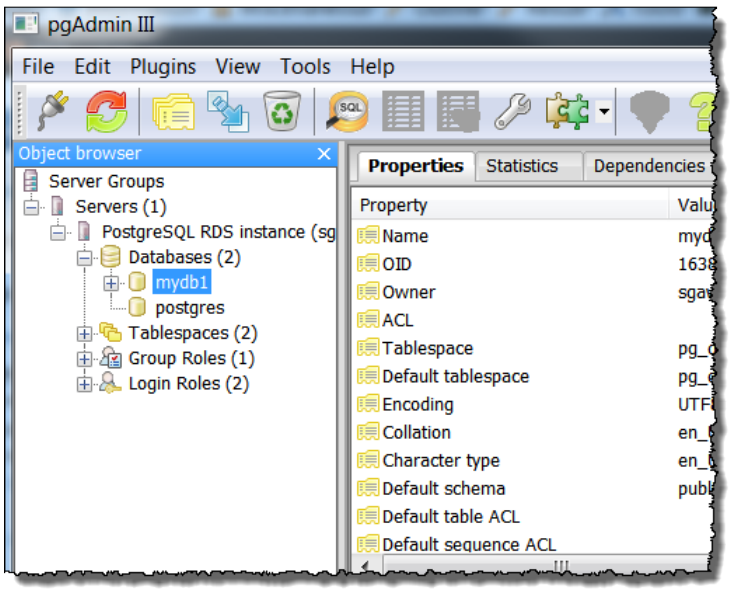
To connect to a PostgreSQL DB instance using pgAdmin

1. Launch the *pgAdmin* application on your client computer. You can install *pgAdmin* from http://www.pgadmin.org/.
2. Select **Add Server** from the **File** menu.
3. In the **New Server Registration** dialog box, enter the DB instance endpoint (for example, mypostgresql.c6c8dntfzzhgv0.us-west-2.rds.amazonaws.com) in the **Host** text box. Do not include the colon or port number as shown on the Amazon RDS console (mypostgresql.c6c8dntfzzhgv0.us-west-2.rds.amazonaws.com*:5432*).

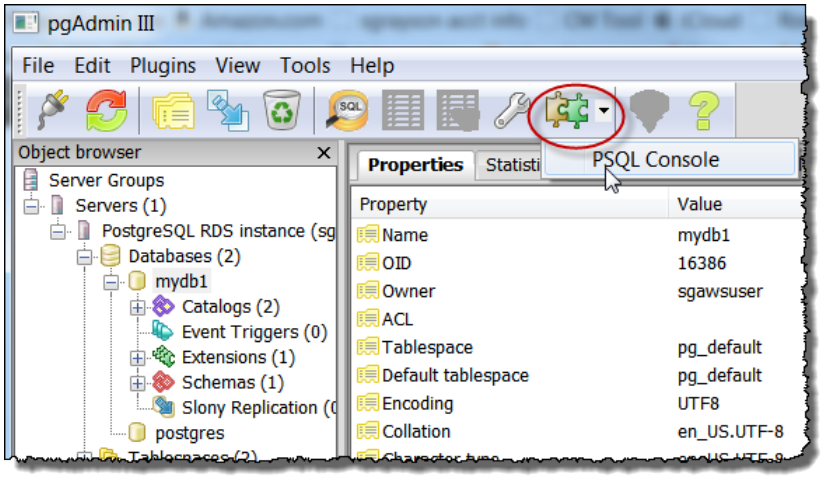
   Enter the port you assigned to the DB instance into the **Port** text box. Enter the user name and user password you entered when you created the DB instance into the **Username** and **Password** text boxes, respectively.



4. Click **OK**.
5. In the **Object browser**, expand the **Server Groups**. Select the Server (the DB instance) you created, and then select the database name.



6. Click the plugin icon and click **PSQL Console**. The *psql* command window opens for the default database you created.



7. Use the command window to enter SQL or *psql* commands. Type `\q` to close the window.

Using *psql* to Connect to a PostgreSQL DB Instance

If your client computer has PostgreSQL installed, you can use a local instance of *psql* to connect to a PostgreSQL DB instance. To connect to your PostgreSQL DB instance using *psql*, you need to provide host information and access credentials.

The following format is used to connect to a PostgreSQL DB instance on Amazon RDS:

```
psql --host=<DB instance endpoint> --port=<port> --username=<master user name>
 --password --dbname=<database name>
```

For example, the following command connects to a database called `mypgdb` on a PostgreSQL DB instance called `mypostgresql` using fictitious credentials:

```
psql --host=mypostgresql.c6c8mwvfdgv0.us-west-2.rds.amazonaws.com --port=5432
--username=awsuser --password --dbname=mypgdb
```

Troubleshooting Connection Issues

By far the most common problem that occurs when attempting to connect to a database on a DB instance is the access rules in the security group assigned to the DB instance. If you used the default DB security group when you created the DB instance, chances are good that the security group did not have the rules

that will allow you to access the instance. For more information about Amazon RDS security groups, see Amazon RDS Security Groups (p. 110)

The most common error is *could not connect to server: Connection timed out*. If you receive this error, check that the host name is the DB instance endpoint and that the port number is correct. Check that the DB security group assigned to the DB instance has the necessary rules to allow access through any firewall your connection may be going through.

Related Topics

- Amazon RDS DB Instances (p. 64)
- Creating a DB Instance Running the PostgreSQL Database Engine (p. 363)
- Amazon RDS Security Groups (p. 110)
- Deleting a DB Instance (p. 459)

Modifying a DB Instance Running the PostgreSQL Database Engine

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. This topic guides you through modifying an Amazon RDS PostgreSQL DB instance, and describes the settings for PostgreSQL instances. For information about additional tasks, such as renaming, rebooting, deleting, tagging, or upgrading an Amazon RDS DB instance, see Amazon RDS DB Instance Lifecycle (p. 440). We recommend that you test any changes on a test instance before modifying a production instance so you better understand the impact of a change. This is especially important when upgrading database versions.

You can have the changes apply immediately or have them applied during the DB instance's next maintenance window. Applying changes immediately can cause an outage in some cases; for more information on the impact of the **Apply Immediately** option when modifying a DB instance, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 455).

AWS Management Console

To modify a PostgreSQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Instances**.
3. Select the check box for the DB instance that you want to change, and then click **Modify**.
4. In the **Modify DB Instance** dialog box, change any of the following settings that you want:

Setting	Description
DB Engine Version	In the list provided, click the version of the PostgreSQL database engine that you want to use.
DB Instance Class	In the list provided, click the DB instance class that you want to use. For information about instance classes, see DB Instance Class (p. 65).
Multi-AZ Deployment	If you want to create a standby replica of your DB instance in another Availability Zone, click **Yes**; otherwise, click **No**. For more information on Multi-AZ deployments, see High Availability (Multi-AZ) (p. 71).
Allocated Storage	Specify how much storage, in gigabytes, to allocate for your DB instance. The minimum allowable value is 5 GB; the maximum is 6 TB. Note that you can only increase the amount of storage when modifying a DB instance, you cannot reduce the amount of storage allocated. For more information on allocated storage, see Amazon RDS Storage Types (p. 77)
Storage Type	Select the storage type you want to use. Changing from **Magnetic** to **General Purpose (SSD)** or **Provisioned IOPS (SSD)** will result in an outage. Also, changing from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic** will result in an outage. For more information about storage, see Storage for Amazon RDS (p. 77).

Setting	Description
DB Instance Identifier	You can rename the DB instance by typing a new name. When you change the DB instance identifier, an instance reboot will occur immediately if you set `Apply Immediately` to true, or will occur during the next maintenance window if you set `Apply Immediately` to false. This value is stored as a lowercase string.
New Master Password	Type a password for your master user. The password must contain from 8 to 41 alphanumeric characters.
Security Groups	Select the security group you want associated with the DB instance. For more information about security groups, see Working with DB Security Groups (p. 537).
Parameter Group	Select the parameter group you want associated with the DB instance. Changing this setting does not result in an outage. The parameter group name itself is changed immediately, but the actual parameter changes are not applied until you reboot the instance without failover. The DB instance will NOT be rebooted automatically and the parameter changes will NOT be applied during the next maintenance window. For more information about parameter groups, see Working with DB Parameter Groups (p. 523).
Option Group	No options are available for PostgreSQL DB instances. For more information about option groups, see Working with Option Groups (p. 510).
Backup Retention Period	Specify the number of days that automatic backups will be retained. To disable automatic backups, set this value to 0. **Note** An immediate outage will occur if you change the backup retention period from 0 to a non-zero value or from a non-zero value to 0.
Backup Window	Set the time range during which automated backups of your databases will occur. Specify a start time in Universal Coordinated Time (UTC) and a duration in hours.
Auto Minor Version Upgrade	If you want your DB instance to receive minor engine version upgrades automatically when they become available, click **Yes**. Upgrades are installed only during your scheduled maintenance window.
Maintenance Window	Set the time range during which system maintenance, including upgrades, will occur. Specify a start time in UTC and a duration in hours.

5. To apply the changes immediately, select the **Apply Immediately** check box. Selecting this option can cause an outage in some cases; for more information on the impact of the **Apply Immediately** option, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 455).
6. When all the changes are as you want them, click **Continue**. If you want to cancel any changes, click the **X** in the upper right corner of the page.

7. Confirm that the changes you want are listed in the summary screen, and then click **Modify DB Instance**.

CLI

To modify a PostgreSQL DB instance

- Use the command `rds-modify-db-instance`.

API

To modify a PostgreSQL DB instance

- Use the `ModifyDBInstance` `action`.

Importing Data into PostgreSQL on Amazon RDS

If you have an existing PostgreSQL deployment that you want to move to Amazon RDS, the complexity of your task depends on the size of your database and the types of database objects that you are transferring. For example, a database that contains data sets on the order of gigabytes, along with stored procedures and triggers, is going to be more complicated than a simple database with only a few megabytes of test data and no triggers or stored procedures.

When loading data into an Amazon RDS PostgreSQL DB instance, you should modify your DB instance settings and your DB parameter group values to allow for the most efficient importing of data into your DB instance.

Modify your DB instance settings to the following:

- Disable DB instance backups (set backup_retention to 0)
- Disable Multi-AZ

Modify your DB parameter group to include the following settings. You should test the parameter settings to find the most efficient settings for your DB instance:

- Increase the value of the `maintenance_work-mem` parameter
- Increase the value of the `checkpoint_segments` and `checkpoint_timeout` parameters to reduce the number of writes to the wal log
- Disable the `synchronous_commit` parameter (do not turn off FSYNC)
- Disable the PostgreSQL autovacuum parameter

Use the `pg_dump -Fc` (compressed) or `pg_restore -j` (parallel) commands with these settings.

> **Note**
> The PostgreSQL command `pg_dumpall` requires super_user permissions that are not granted when you create a DB instance, so it cannot be used for importing data.

Importing a PostgreSQL Database from an Amazon EC2 Instance

If you have data in a PostgreSQL server on an Amazon EC2 instance and want to move it to a PostgreSQL DB instance, you can use the following process. The following list shows the steps to take. Each step is discussed in more detail in the following sections.

1. Create a file using pg_dump that contains the data to be loaded
2. Create the target DB instance
3. Use *psql* to create the database on the DB instance and load the data
4. Create a DB snapshot of the DB instance

Step 1: Create a file using `pg_dump` that contains the data to be loaded

`pg_dump` uses the COPY command to create a schema and data dump of a PostgreSQL database. The dump script generated by `pg_dump` loads data into a database with the same name and recreates the tables, indexes, and foreign keys. You can use the `pg_restore` command and the `-d` parameter to restore the data to a database with a different name.

Before you create the data dump, you should query the tables to be dumped to get a row count so you can confirm the count on the target DB instance.

The following command creates a dump file called mydb2dump.sql for a database called mydb2.

```
prompt>pg_dump dbname=mydb2 -f mydb2dump.sql
```

Step 2: Create the target DB instance

Create the target PostgreSQL DB instance using either the Amazon RDS console, CLI, or API. Create the instance with the backup retention setting set to 0 and disable Multi-AZ. This will allow faster data import. You must create a database on the instance before you can dump the data. The database can have the same name as the database that is contained the dumped data or you can create a database with a different name and use the `pg_restore` command and the `-d` parameter to restore the data into the newly named database.

Step 3: Use *psql* to create the database on the DB instance and load the data

You can use the same connection you used to execute the pg_dump command to connect to the target DB instance and recreate the database. Using *psql*, you can use the master user name and master password to create the database on the DB instance

The following example uses *psql* and a dump file named mydb2dump.sql to create a database called mydb2 on a PostgreSQL DB instance called mypginstance:

```
 psql -f mydb2dump.sql --host=mypginstance.c6c8mntzhgv0.us-west-2.rds.amazon
aws.com
--port=8199 --username=myawsuser --password --dbname=mydb2
```

Step 4: Create a DB snapshot of the DB instance

Once you have verified that the data was loaded into your DB instance, we recommend that you create a DB snapshot of the target PostgreSQL DB instance. DB snapshots are complete backups of your DB instance that can be used to restore your DB instance to a known state. A DB snapshot taken immediately after the load protects you from having to load the data again in case of a mishap and can also be used to seed new database instances. For information about creating a DB snapshot, see Creating a DB Snapshot (p. 499).

Using the `\copy` Command to Import Data to a Table on a PostgreSQL DB Instance

You can run the `\copy` command from the *psql* prompt to import data into a table on a PostgreSQL DB instance. The table must already exist on the DB instance.

> **Note**
> The `\copy` command does not provide confirmation of actions, such as a count of rows inserted. PostgreSQL does provide error messages if the copy command fails due to an error.

Create a .csv file from the data in the source table, log on to the target database on the PostgreSQL instance using *psql*, and then run the following command. This example uses *source-table* a+s the source table name, *source-table.csv* as the .csv file, and *target-db* as the target database:

```
target-db=> \copy source-table from 'source-table.csv' with DELIMITER ',';
```

You can also run the following command from your client computer command prompt. This example uses *source-table* as the source table name, *source-table.csv* as the .csv file, and *target-db* as the target database:

```
$psql target-db -U <admin user> -p <port> -h <DB instance name> -c "\copy source-
table from 'source-table.csv' with DELIMITER ','"
```

Appendix: Common DBA Tasks for PostgreSQL

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances running the PostgreSQL database engine. In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.

For information about working with PostgreSQL log files on Amazon RDS, see PostgreSQL Database Log Files (p. 603)

Topics

- Creating Roles (p. 380)
- Managing PostgreSQL Database Access (p. 380)
- Working with PostgreSQL Parameters (p. 381)
- Setting up PostGIS (p. 389)
- Using pgBadger for Log Analysis with PostgreSQL (p. 391)

Creating Roles

When you create a DB instance, the master user system account that you create is assigned to the `rds_superuser` role. The `rds_superuser` role is a pre-defined Amazon RDS role similar to the PostgreSQL superuser role (customarily named postgres in local instances), but with some restrictions. As with the PostgreSQL superuser role, the `rds_superuser` role has the most privileges on your DB instance and you should not assign this role to users unless they need the most access to the DB instance.

The following example shows how to create a user and then grant the user the `rds_superuser` role. User-defined roles, such as `rds_superuser`, have to be granted.

```
postgres=> create role testuser with password 'testuser' login;
CREATE ROLE
postgres=> grant rds_superuser to testuser;
GRANT ROLE
postgres=>
```

Managing PostgreSQL Database Access

By default, when PostgreSQL database objects are created, they receive "public" access privileges. You can revoke all privileges to a database and then explicitly add privileges back as you need them.

As the master user, you can remove all privileges from a database using the following command format:

```
postgres=> revoke all on database <database name> from public;
REVOKE
```

You can then add privileges back to a user. For example, the following command grants connect access to a user named *mytestuser* to a database named *test*.

```
test=> grant connect on database test to mytestuser;
GRANT
```

Note that on a local instance, you could specify database privileges in the pg_hba.conf file, but when using PostgreSQL with Amazon RDS it is better to restrict privileges at the Postgres level. Changes to the pg_hba.conf file require a server restart so you cannot edit the pg_hba.conf in Amazon RDS, but privilege changes at the Postgres level occur immediately.

Working with PostgreSQL Parameters

PostgreSQL parameters that you would set for a local PostgreSQL instance in the *postgresql.conf* file are maintained in the DB parameter group for your DB instance. If you create a DB instance using the default parameter group, the parameter settings are in the parameter group called *default.postgres9.3*.

When you create a DB instance, the parameters in the associated DB parameter group are loaded. You can modify parameter values by changing values in the parameter group. You can also change parameter values, if you have the security privileges to do so, by using the ALTER DATABASE, ALTER ROLE, and the SET commands. Note that you cannot use the command line `postgres` command nor the `env PGOPTIONS` command because you will have no access to the host.

Keeping track of PostgreSQL parameter settings can occasionally be difficult. Use the following command to list current parameter settings and the default value:

```
select name, setting, boot_val, reset_val, unit
from pg_settings
order by name;
```

For an explanation of the output values, see the `pg_settings` topic in the PostgreSQL documentation.

If you set the memory settings too large for `max_connections`, `shared_buffers`, or `effective_cache_size`, you will prevent the PostgreSQL instance from starting up. Note that some parameters use units that you might not be familiar with; for example, `shared_buffers` sets the number of 8 KB shared memory buffers used by the server.

The following error is written to the *postgres.log* file when the instance is attempting to start up, but incorrect parameter settings are preventing it from starting.

```
2013-09-18 21:13:15 UTC::@:[8097]:FATAL:  could not map anonymous shared
memory: Cannot allocate memory
2013-09-18 21:13:15 UTC::@:[8097]:HINT:  This error usually means that
PostgreSQL's request for a shared memory segment exceeded available memory or
swap space. To reduce the request size (currently 3514134274048 bytes), reduce

PostgreSQL's shared memory usage, perhaps by reducing shared_buffers or
max_connections.
```

There are two types of PostgreSQL parameters, fixed and dynamic. Fixed parameters require that the DB instance be rebooted before they are applied. Dynamic parameters can be applied immediately. The following table shows parameters you can modify for a PostgreSQL DB instance and the parameter's type:

Parameter Name	Apply_Type	Description
application_name	Dynamic	Sets the application name to be reported in statistics and logs.
array_nulls	Dynamic	Enables input of NULL elements in arrays.
authentication_timeout	Dynamic	Sets the maximum allowed time to complete client authentication.
autovacuum	Dynamic	Starts the autovacuum subprocess.
autovacuum_analyze_scale_factor	Dynamic	Number of tuple inserts, updates, or deletes prior to analyze as a fraction of reltuples.
autovacuum_analyze_threshold	Dynamic	Minimum number of tuple inserts, updates, or deletes prior to analyze.
autovacuum_naptime	Dynamic	Time to sleep between autovacuum runs.
autovacuum_vacuum_cost_delay	Dynamic	Vacuum cost delay, in milliseconds, for autovacuum.
autovacuum_vacuum_cost_limit	Dynamic	Vacuum cost amount available before napping, for autovacuum.
autovacuum_vacuum_scale_factor	Dynamic	Number of tuple updates or deletes prior to vacuum as a fraction of reltuples.
autovacuum_vacuum_threshold	Dynamic	Minimum number of tuple updates or deletes prior to vacuum.
backslash_quote	Dynamic	Sets whether a backslash (\) is allowed in string literals.
bgwriter_delay	Dynamic	Background writer sleep time between rounds.
bgwriter_lru_maxpages	Dynamic	Background writer maximum number of LRU pages to flush per round.
bgwriter_lru_multiplier	Dynamic	Multiple of the average buffer usage to free per round.
bytea_output	Dynamic	Sets the output format for bytea.
check_function_bodies	Dynamic	Checks function bodies during CREATE FUNCTION.
checkpoint_completion_target	Dynamic	Time spent flushing dirty buffers during checkpoint, as fraction of checkpoint interval.
checkpoint_segments	Dynamic	Sets the maximum distance in log segments between automatic WAL checkpoints.
checkpoint_timeout	Dynamic	Sets the maximum time between automatic WAL checkpoints.
checkpoint_warning	Dynamic	Enables warnings if checkpoint segments are filled more frequently than this.
client_encoding	Dynamic	Sets the client's character set encoding.
client_min_messages	Dynamic	Sets the message levels that are sent to the client.

Parameter Name	Apply_Type	Description
commit_delay	Dynamic	Sets the delay in microseconds between transaction commit and flushing WAL to disk.
commit_siblings	Dynamic	Sets the minimum concurrent open transactions before performing commit_delay.
constraint_exclusion	Dynamic	Enables the planner to use constraints to optimize queries.
cpu_index_tuple_cost	Dynamic	Sets the planner's estimate of the cost of processing each index entry during an index scan.
cpu_operator_cost	Dynamic	Sets the planner's estimate of the cost of processing each operator or function call.
cpu_tuple_cost	Dynamic	Sets the planner's estimate of the cost of processing each tuple (row).
cursor_tuple_fraction	Dynamic	Sets the planner's estimate of the fraction of a cursor's rows that will be retrieved.
datestyle	Dynamic	Sets the display format for date and time values.
deadlock_timeout	Dynamic	Sets the time to wait on a lock before checking for deadlock.
debug_pretty_print	Dynamic	Indents parse and plan tree displays.
debug_print_parse	Dynamic	Logs each query's parse tree.
debug_print_plan	Dynamic	Logs each query's execution plan.
debug_print_rewritten	Dynamic	Logs each query's rewritten parse tree.
default_statistics_target	Dynamic	Sets the default statistics target.
default_tablespace	Dynamic	Sets the default tablespace to create tables and indexes in.
default_transaction_deferrable	Dynamic	Sets the default deferrable status of new transactions.
default_transaction_isolation	Dynamic	Sets the transaction isolation level of each new transaction.
default_transaction_read_only	Dynamic	Sets the default read-only status of new transactions.
default_with_oids	Dynamic	Creates new tables with OIDs by default.
effective_cache_size	Dynamic	Sets the planner's assumption about the size of the disk cache.
effective_io_concurrency	Dynamic	Number of simultaneous requests that can be handled efficiently by the disk subsystem.
enable_bitmapscan	Dynamic	Enables the planner's use of bitmap-scan plans.
enable_hashagg	Dynamic	Enables the planner's use of hashed aggregation plans.

Parameter Name	Apply_Type	Description
enable_hashjoin	Dynamic	Enables the planner's use of hash join plans.
enable_indexscan	Dynamic	Enables the planner's use of index-scan plans.
enable_material	Dynamic	Enables the planner's use of materialization.
enable_mergejoin	Dynamic	Enables the planner's use of merge join plans.
enable_nestloop	Dynamic	Enables the planner's use of nested-loop join plans.
enable_seqscan	Dynamic	Enables the planner's use of sequential-scan plans.
enable_sort	Dynamic	Enables the planner's use of explicit sort steps.
enable_tidscan	Dynamic	Enables the planner's use of TID scan plans.
escape_string_warning	Dynamic	Warns about backslash (\) escapes in ordinary string literals.
extra_float_digits	Dynamic	Sets the number of digits displayed for floating-point values.
from_collapse_limit	Dynamic	Sets the FROM-list size beyond which subqueries are not collapsed.
fsync	Dynamic	Forces synchronization of updates to disk.
full_page_writes	Dynamic	Writes full pages to WAL when first modified after a checkpoint.
geqo	Dynamic	Enables genetic query optimization.
geqo_effort	Dynamic	GEQO: effort is used to set the default for other GEQO parameters.
geqo_generations	Dynamic	GEQO: number of iterations of the algorithm.
geqo_pool_size	Dynamic	GEQO: number of individuals in the population.
geqo_seed	Dynamic	GEQO: seed for random path selection.
geqo_selection_bias	Dynamic	GEQO: selective pressure within the population.
geqo_threshold	Dynamic	Sets the threshold of FROM items beyond which GEQO is used.
gin_fuzzy_search_limit	Dynamic	Sets the maximum allowed result for exact search by GIN.
intervalstyle	Dynamic	Sets the display format for interval values.
join_collapse_limit	Dynamic	Sets the FROM-list size beyond which JOIN constructs are not flattened.
lc_messages	Dynamic	Sets the language in which messages are displayed.
lc_monetary	Dynamic	Sets the locale for formatting monetary amounts.
lc_numeric	Dynamic	Sets the locale for formatting numbers.

Parameter Name	Apply_Type	Description
lc_time	Dynamic	Sets the locale for formatting date and time values.
log_autovacuum_min_duration	Dynamic	Sets the minimum execution time above which autovacuum actions will be logged.
log_checkpoints	Dynamic	Logs each checkpoint.
log_connections	Dynamic	Logs each successful connection.
log_disconnections	Dynamic	Logs end of a session, including duration.
log_duration	Dynamic	Logs the duration of each completed SQL statement.
log_error_verbosity	Dynamic	Sets the verbosity of logged messages.
log_executor_stats	Dynamic	Writes executor performance statistics to the server log.
log_filename	Dynamic	Sets the file name pattern for log files.
log_hostname	Dynamic	Logs the host name in the connection logs.
log_lock_waits	Dynamic	Logs long lock waits.
log_min_duration_statement	Dynamic	Sets the minimum execution time above which statements will be logged.
log_min_error_statement	Dynamic	Causes all statements generating an error at or above this level to be logged.
log_min_messages	Dynamic	Sets the message levels that are logged.
log_parser_stats	Dynamic	Writes parser performance statistics to the server log.
log_planner_stats	Dynamic	Writes planner performance statistics to the server log.
log_rotation_age	Dynamic	Automatic log file rotation will occur after N minutes.
log_rotation_size	Dynamic	Automatic log file rotation will occur after N kilobytes.
log_statement	Dynamic	Sets the type of statements logged.
log_statement_stats	Dynamic	Writes cumulative performance statistics to the server log.
log_temp_files	Dynamic	Logs the use of temporary files larger than this number of kilobytes.
maintenance_work_mem	Dynamic	Sets the maximum memory to be used for maintenance operations.
max_stack_depth	Dynamic	Sets the maximum stack depth, in kilobytes.
max_standby_archive_delay	Dynamic	Sets the maximum delay before canceling queries when a hot standby server is processing archived WAL data.
max_standby_streaming_delay	Dynamic	Sets the maximum delay before canceling queries when a hot standby server is processing streamed WAL data.

Parameter Name	Apply_Type	Description
quote_all_identifiers	Dynamic	Adds quotes (") to all identifiers when generating SQL fragments.
random_page_cost	Dynamic	Sets the planner's estimate of the cost of a non-sequentially fetched disk page.
rds.log_retention_period	Dynamic	Amazon RDS will delete PostgreSQL logs that are older than N minutes.
search_path	Dynamic	Sets the schema search order for names that are not schema-qualified.
seq_page_cost	Dynamic	Sets the planner's estimate of the cost of a sequentially fetched disk page.
session_replication_role	Dynamic	Sets the sessions behavior for triggers and rewrite rules.
sql_inheritance	Dynamic	Causes subtables to be included by default in various commands.
ssl_renegotiation_limit	Dynamic	Sets the amount of traffic to send and receive before renegotiating the encryption keys.
standard_conforming_strings	Dynamic	Causes ... strings to treat backslashes literally.
statement_timeout	Dynamic	Sets the maximum allowed duration of any statement.
synchronize_seqscans	Dynamic	Enables synchronized sequential scans.
synchronous_commit	Dynamic	Sets the current transactions synchronization level.
tcp_keepalives_count	Dynamic	Maximum number of TCP keepalive retransmits.
tcp_keepalives_idle	Dynamic	Time between issuing TCP keepalives.
tcp_keepalives_interval	Dynamic	Time between TCP keepalive retransmits.
temp_buffers	Dynamic	Sets the maximum number of temporary buffers used by each session.
temp_tablespaces	Dynamic	Sets the tablespaces to use for temporary tables and sort files.
timezone	Dynamic	Sets the time zone for displaying and interpreting time stamps.
track_activities	Dynamic	Collects information about executing commands.
track_counts	Dynamic	Collects statistics on database activity.
track_functions	Dynamic	Collects function-level statistics on database activity.
track_io_timing	Dynamic	Collects timing statistics on database I/O activity.
transaction_deferrable	Dynamic	Indicates whether to defer a read-only serializable transaction until it can be executed with no possible serialization failures.
transaction_isolation	Dynamic	Sets the current transactions isolation level.

Parameter Name	Apply_Type	Description
transaction_read_only	Dynamic	Sets the current transactions read-only status.
transform_null_equals	Dynamic	Treats expr=NULL as expr IS NULL.
update_process_title	Dynamic	Updates the process title to show the active SQL command.
vacuum_cost_delay	Dynamic	Vacuum cost delay in milliseconds.
vacuum_cost_limit	Dynamic	Vacuum cost amount available before napping.
vacuum_cost_page_dirty	Dynamic	Vacuum cost for a page dirtied by vacuum.
vacuum_cost_page_hit	Dynamic	Vacuum cost for a page found in the buffer cache.
vacuum_cost_page_miss	Dynamic	Vacuum cost for a page not found in the buffer cache.
vacuum_defer_cleanup_age	Dynamic	Number of transactions by which vacuum and hot cleanup should be deferred, if any.
vacuum_freeze_min_age	Dynamic	Minimum age at which vacuum should freeze a table row.
vacuum_freeze_table_age	Dynamic	Age at which vacuum should scan a whole table to freeze tuples.
wal_writer_delay	Dynamic	WAL writer sleep time between WAL flushes.
work_mem	Dynamic	Sets the maximum memory to be used for query workspaces.
xmlbinary	Dynamic	Sets how binary values are to be encoded in XML.
xmloption	Dynamic	Sets whether XML data in implicit parsing and serialization operations is to be considered as documents or content fragments.
autovacuum_freeze_max_age	Static	Age at which to autovacuum a table to prevent transaction ID wraparound.
autovacuum_max_workers	Static	Sets the maximum number of simultaneously running autovacuum worker processes.
max_connections	Static	Sets the maximum number of concurrent connections.
max_files_per_process	Static	Sets the maximum number of simultaneously open files for each server process.
max_locks_per_transaction	Static	Sets the maximum number of locks per transaction.
max_pred_locks_per_transaction	Static	Sets the maximum number of predicate locks per transaction.
max_prepared_transactions	Static	Sets the maximum number of simultaneously prepared transactions.
shared_buffers	Static	Sets the number of shared memory buffers used by the server.
ssl	Static	Enables SSL connections.

Parameter Name	Apply_Type	Description
track_activity_query_size	Static	Sets the size reserved for pg_stat_activity.current_query, in bytes.
wal_buffers	Static	Sets the number of disk-page buffers in shared memory for WAL.

Amazon RDS uses the default PostgreSQL units for all parameters. The following table shows the PostgreSQL unit value for each parameter.

Parameter Name	Unit
effective_cache_size	8 KB
segment_size	8 KB
shared_buffers	8 KB
temp_buffers	8 KB
wal_buffers	8 KB
wal_segment_size	8 KB
log_rotation_size	KB
log_temp_files	KB
maintenance_work_mem	KB
max_stack_depth	KB
ssl_renegotiation_limit	KB
temp_file_limit	KB
work_mem	KB
log_rotation_age	min
autovacuum_vacuum_cost_delay	ms
bgwriter_delay	ms
deadlock_timeout	ms
lock_timeout	ms
log_autovacuum_min_duration	ms
log_min_duration_statement	ms
max_standby_archive_delay	ms
max_standby_streaming_delay	ms
statement_timeout	ms
vacuum_cost_delay	ms

Parameter Name	Unit
wal_receiver_timeout	ms
wal_sender_timeout	ms
wal_writer_delay	ms
archive_timeout	s
authentication_timeout	s
autovacuum_naptime	s
checkpoint_timeout	s
checkpoint_warning	s
post_auth_delay	s
pre_auth_delay	s
tcp_keepalives_idle	s
tcp_keepalives_interval	s
wal_receiver_status_interval	s

Setting up PostGIS

There is a bit of setup you need to do before you can use the PostGIS extension. The following list shows what you need to do. Each step is described in greater detail in this section.

- Connect to the DB instance using the master username used to create the DB instance
- Load the PostGIS extensions
- Transfer ownership of the extensions to the `rds_superuser` role
- Transfer ownership of the objects to the `rds_superuser` role
- Test the extensions

Step 1: Connect to the DB instance using the master username used to create the DB instance

The master username you used to create the DB instance is automatically assigned the `rds_superuser` role. When you connect to the DB instance, the you will be in the `rds_superuser` role that is needed to do the remaining steps.

The following example uses SELECT to show you the current user; in this case, the current user should be the master username you chose when creating the DB instance:

```
mydb1=> select current_user;
 current_user
 -------------
  myawsuser
```

```
 (1 row)
```

Step 2: Load the PostGIS extensions

Use the CREATE EXTENSION statements to load the PostGIS extensions. Note that you must also load the `fuzzystrmatch` extension. You can then use the `\dn` *psql* command to list the owners of the PostGIS schemas.

```
mydb1=> create extension postgis;
CREATE EXTENSION
mydb1=> create extension fuzzystrmatch;
CREATE EXTENSION
mydb1=> create extension postgis_tiger_geocoder;
CREATE EXTENSION
mydb1=> create extension postgis_topology;
CREATE EXTENSION
mydb1=> \dn
     List of schemas
     Name     |   Owner
--------------+-----------
 public       | myawsuser
 tiger        | rdsadmin
 topology     | rdsadmin
(4 rows)
```

Step 3: Transfer ownership of the extensions to the rds_superuser role

Use the ALTER SCHEMA statements to transfer ownership of the schemas to the `rds_superuser` role.

```
mydb1=> alter schema tiger owner to rds_superuser;
ALTER SCHEMA
mydb1=> alter schema topology owner to rds_superuser;
ALTER SCHEMA
mydb1=> \dn
       List of schemas
     Name     |     Owner
--------------+---------------
 public       | myawsuser
 tiger        | rds_superuser
 topology     | rds_superuser
(4 rows)
```

Step 4: Transfer ownership of the objects to the rds_superuser role

Use the following function to transfer ownership of the PostGIS objects to the `rds_superuser` role. You can run the function from the *psql* prompt:

```
CREATE FUNCTION exec(text) returns text language plpgsql volatile AS $f$ BEGIN
 EXECUTE $1; RETURN $1; END; $f$;
SELECT exec('ALTER TABLE ' || quote_ident(s.nspname) || '.' ||
quote_ident(s.relname) || ' OWNER TO rds_superuser')
  FROM (
    SELECT nspname, relname
    FROM pg_class c JOIN pg_namespace n ON (c.relnamespace = n.oid)
    WHERE nspname in ('tiger','topology') AND
    relkind IN ('r','S','v') ORDER BY relkind = 'S')
s;
```

Step 5: Test the extensions

Test `tiger` by using the following SELECT statement:

```
mydb1=> select na.address, na.streetname, na.streettypeabbrev, na.zip
mydb1-> from normalize_address('1 Devonshire Place, Boston, MA 02109') as na;
 address | streetname | streettypeabbrev |  zip
---------+------------+------------------+-------
       1 | Devonshire | Pl               | 02109
(1 row)
```

Test `topology` by using the following SELECT statement:

```
mydb1=> select topology.createtopology('my_new_topo',26986,0.5);
 createtopology
----------------
              1
(1 row)
```

Using pgBadger for Log Analysis with PostgreSQL

You can use a log analyzer such as pgbadger to analyze PostgreSQL logs. Although the *pgbadger* documentation states that the %l pattern (log line for session/process) should be a part of the prefix, if you provide the current rds log_line_prefix as a parameter to *pgbadger* it should still produce a report.

For example, the following command would correctly format an Amazon RDS PostgreSQL log file dated 2014-02-04 using *pgbadger*:

```
./pgbadger -p '%t:%r:%u@%d:[%p]:' postgresql.log.2014-02-04-00
```

Aurora on Amazon RDS

Amazon Aurora is a fully managed, MySQL-compatible, relational database engine that combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. It delivers up to five times the performance of MySQL without requiring changes to most of your existing applications.

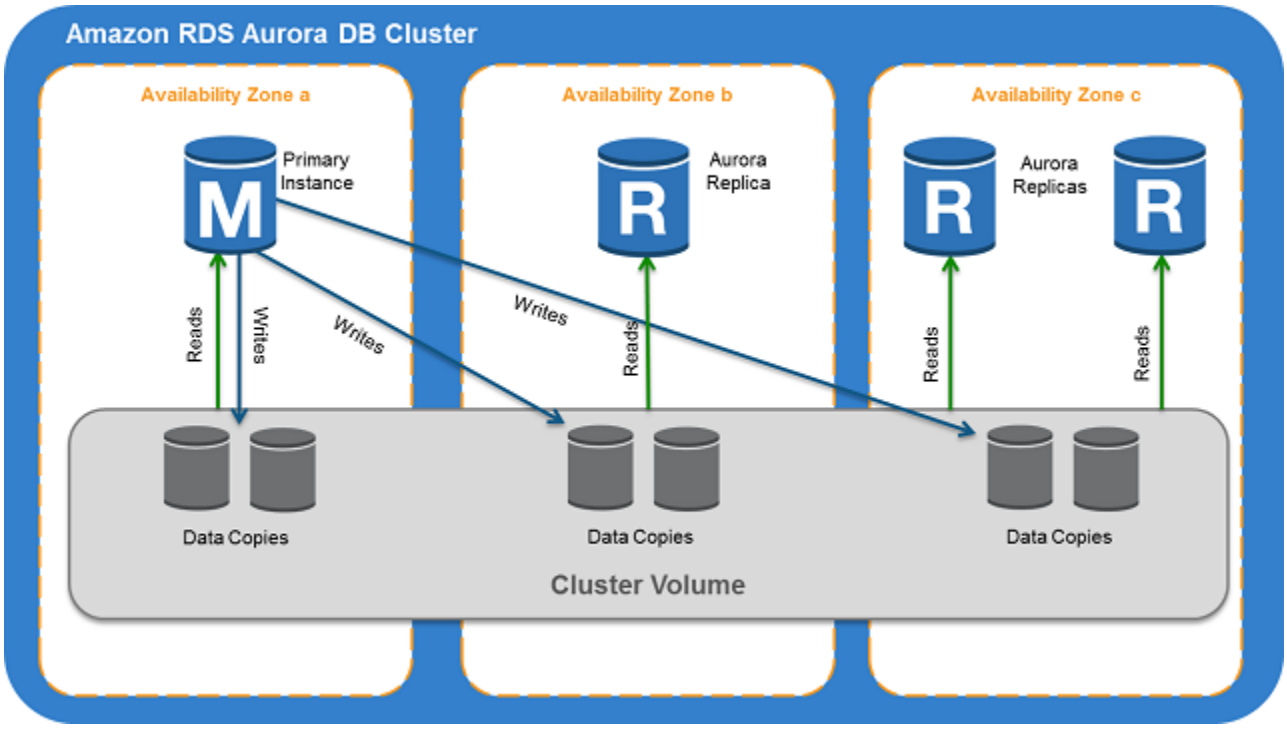
Amazon Aurora makes it simple and cost-effective to set up, operate, and scale your new and existing MySQL deployments, thus freeing you to focus on your business and applications. Amazon RDS provides administration for Amazon Aurora by handling routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair. Amazon RDS also provides push-button migration tools to convert your existing Amazon RDS for MySQL applications to Amazon Aurora.

Amazon Aurora is a drop-in replacement for MySQL. The code, tools and applications you use today with your existing MySQL databases can be used with Amazon Aurora.

When you create an Amazon Aurora instance, you create a *DB cluster*. A DB cluster consists of one or more instances, and a cluster volume that manages the data for those instances. An Aurora *cluster volume* is a virtual database storage volume that spans multiple Availability Zones, with each Availability Zone having a copy of the cluster data. Two types of instances make up an Aurora DB cluster:

- **Primary instance** – Supports read-write workloads, and performs all of the data modifications to the cluster volume. Each Aurora DB cluster has one primary instance.
- **Aurora Replica** – Supports only read operations. Each DB cluster can have up to 15 Aurora Replicas in addition to the primary instance, which supports both read and write workloads. Multiple Aurora Replicas distribute the read workload, and by locating Aurora Replicas in separate Availability Zones you can also increase database availability.

The following diagram illustrates the relationship between the Amazon Aurora cluster volume and the primary and Aurora Replicas in the Aurora DB cluster.



Availability

The following table shows the regions where Amazon Aurora is currently available.

Region	Console link
US East (N. Virginia)	https://console.amazonaws.cn/rds/home?region=us-east-1
US West (Oregon)	https://console.amazonaws.cn/rds/home?region=us-west-2
EU (Ireland)	https://console.amazonaws.cn/rds/home?region=eu-west-1

Aurora Endpoints

Each Aurora DB cluster has a cluster endpoint that you can connect to. An endpoint is made up of a domain name and a port separated by a colon, for example: `mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com:3306`.

The cluster endpoint connects you to the primary instance for the DB cluster. You can perform both read and write operations using the cluster endpoint. The primary instance also has a unique endpoint. The difference between the two endpoints is that the cluster endpoint will always point to the primary instance. If the primary instance fails, then the cluster endpoint will point to the new primary instance. For more information on failovers, see Fault Tolerance for an Aurora DB Cluster (p. 430).

Each Aurora Replica in an Aurora DB cluster has a unique endpoint. You can configure multiple clients to connect to different Aurora Replicas in an Aurora DB cluster to distribute the read workload for your application. For high-availability scenarios, you can also place Aurora Replicas in separate Availability

Zones, which ensures that your application will still be able to read data from your Aurora DB cluster in the event of an Availability Zone failure.

You must connect to the cluster endpoint for high-availability scenarios. This connection ensures that you will continue to have access to the Aurora DB cluster in the event of a failover. During a failover, Aurora continues to serve requests, with minimal interruption of service, to the cluster endpoint from any available instances as it replaces the failed instance.

Consider an Amazon Aurora DB cluster that has two Aurora Replicas in different Availability Zones from its primary instance. By connecting to the cluster endpoint, you can send both read and write traffic to the primary instance. You can also connect to the endpoint for each Aurora Replica and send queries directly to those DB instances. In the unlikely event that the primary instance or the Availability Zone that contains the primary instance fails, then RDS will promote one of the Aurora Replicas to be the new primary instance and update the DNS record for the cluster endpoint to point to the new primary instance. Your application will continue to send read and write traffic to your Aurora DB cluster by using the cluster endpoint with minimal interruption in service.

Amazon Aurora Storage

Aurora data is stored in the cluster volume, which is a single, virtual volume that utilizes solid state disk (SSD) drives. A cluster volume consists of copies of the data across multiple Availability Zones in a single region. Because the data is automatically replicated across Availability Zones, your data is highly durable with less possibility of data loss. This replication also ensures that your database is more available during a failover because the data copies already exist in the other Availability Zones and continue to serve data requests to the instances in your DB cluster.

Aurora cluster volumes automatically grow as the amount of data in your database increases. An Aurora cluster volume can grow to a maximum size of 64 terabytes (TB). Even though an Aurora cluster volume can grow to up to 64 TB, you are only charged for the space that you use in an Aurora cluster volume. For pricing information, go to the Amazon RDS product page.

Amazon Aurora Replication

Aurora Replicas are independent endpoints in an Aurora DB cluster. They provide read-only access to the data in the DB cluster volume and enable you to scale the read workload for your data over multiple replicated instances to both improve the performance of data reads as well as increase the availability of the data in your Aurora DB cluster. Aurora Replicas are also failover targets and are quickly promoted if the primary instance for your Aurora DB cluster fails.

For more information on Aurora Replicas and other options for replicating data in an Aurora DB cluster, see Replication with Amazon Aurora (p. 422).

Amazon Aurora Reliability

Aurora is designed to be reliable, durable, and fault tolerant. You can architect your Aurora DB cluster to improve availability by doing things such as adding Aurora Replicas and placing them in different Availability Zones, and also Aurora includes several automatic features that make it a reliable database solution.

Storage Auto-Repair

Because Aurora maintains multiple copies of your data in three Availability Zones, the chance of losing data as a result of a disk failure is greatly minimized. Aurora automatically detects failures in the disk

volumes that make up the cluster volume. When a segment of a disk volume fails, Aurora immediately repairs the segment. When Aurora repairs the disk segment, it uses the data in the other volumes that make up the cluster volume to ensure that the data in the repaired segment is current. As a result, Aurora avoids data loss and reduces the need to perform a point-in-time restore to recover from a disk failure.

"Survivable" Cache Warming

Aurora "warms" the buffer pool cache when a database starts up after it has been shut down or restarted after a failure. That is, Aurora preloads the buffer pool with the pages for known common queries that are stored in an in-memory page cache. This provides a performance gain by bypassing the need for the buffer pool to "warm up" from normal database use.

The Aurora page cache is managed in a separate process from the database, which allows the page cache to "survive" independently of the database. In the unlikely event of a database failure, the page cache remains in memory, which ensures that the buffer pool is warmed with the most current state when the database restarts.

Crash Recovery

Aurora is designed to recover from a crash almost instantaneously and continue to serve your application data. Aurora performs crash recovery asynchronously on parallel threads, so that your database is open and available immediately after a crash. For more information, see Fault Tolerance for an Aurora DB Cluster (p. 430).

Amazon RDS for Aurora Security

Security for Amazon Aurora is managed at three levels:

- To control who can perform Amazon RDS management actions on Aurora DB clusters and DB instances, you use AWS Identity and Access Management (IAM). When you connect to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS management operations. For more information, see Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources (p. 88).

  If you are using an IAM account to access the Amazon Aurora console, you must first log on to the AWS Management Console with your IAM account, and then go to the Aurora console at https://console.aws.amazon.com/rds.
- Aurora DB clusters must be created in an Amazon Virtual Private Cloud (VPC). To control which devices and Amazon EC2 instances can open connections to the endpoint and port of the DB instance for Aurora DB clusters in a VPC, you use a VPC security group. These endpoint and port connections can be made using Secure Sockets Layer (SSL). In addition, firewall rules at your company can control whether devices running at your company can open connections to a DB instance. For more information on VPCs, see Using Amazon RDS with Amazon Virtual Private Cloud (VPC) (p. 562).
- To authenticate login and permissions for an Amazon Aurora DB instance once a connection has been opened, you take the same approach as with a stand-alone instance of MySQL. Commands such as `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE`, and `SET PASSWORD` work just as they do in on-premises databases, as does directly modifying database schema tables. For information, go to MySQL User Account Management in the MySQL documentation.

When you create an Amazon Aurora DB instance, the master user has the following default privileges:

- `alter`
- `alter routine`

- create
- create routine
- create temporary tables
- create user
- create view
- delete
- drop
- event
- execute
- grant option
- index
- insert
- lock tables
- process
- references
- replication slave
- select
- show databases
- show view
- trigger
- update

To provide management services for each DB cluster, the `rdsadmin` user is created when the DB cluster is created. Attempting to drop, rename, change the password, or change privileges for the `rdsadmin` account will result in an error.

For management of the DB cluster, the standard `kill` and `kill_query` commands have been restricted. Instead, use the Amazon RDS commands `rds_kill` and `rds_kill_query` to terminate user sessions or queries on DB instances.

Securing Aurora Data with SSL

Amazon Aurora DB clusters support Secure Sockets Layer (SSL) connections from applications using the same process and public key as Amazon RDS MySQL DB instances.

Amazon RDS creates an SSL certificate and installs the certificate on the DB instance when Amazon RDS provisions the instance. These certificates are signed by a certificate authority. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. The public key is stored at https://rds.amazonaws.com/doc/rds-ssl-ca-cert.pem.

To encrypt connections using the default **mysql** client, launch the mysql client using the *`--ssl-ca`* parameter to reference the public key, for example:

```
mysql -h mycluster.cluster-c9akciq32.rds-us-east-1.amazonaws.com
--ssl-ca=rds-ssl-ca-cert.pem --ssl-verify-server-cert
```

You can use the GRANT statement to require SSL connections for specific users accounts. For example, you can use the following statement to require SSL connections on the user account `encrypted_user`:

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL
```

Note
For more information on SSL connections with MySQL, go to the MySQL documentation.

Using the memcached Option with Amazon Aurora

Amazon Aurora DB instances support the MySQL 5.6 memcached option, a simple, key-based cache. For more information about the MySQL memcached option, see MySQL 5.6 memcached Support (p. 173).

Important
The `memcached` option is currently not supported for Aurora and will be added at a later time.

Comparison of Amazon RDS for Aurora and Amazon RDS for MySQL

Although Aurora instances are compatible with MySQL client applications, Aurora has advantages over MySQL as well as limitations to the MySQL features that Aurora supports. This functionality can influence your decision about whether Amazon Aurora or MySQL on Amazon RDS are the best cloud database for your solution. The following table shows the differences between Amazon RDS for Aurora and Amazon RDS for MySQL.

Feature	Amazon RDS for Aurora	Amazon RDS for MySQL
Read scaling	Supports up to 15 Aurora Replicas with minimal impact on the performance of write operations.	Supports up to 5 Read Replicas with some impact on the performance of write operations.
Failover target	Aurora Replicas are automatic failover targets with no data loss.	Read Replicas can be manually promoted to the master DB instance with potential data loss.
MySQL version	Supports only MySQL version 5.6.	Supports MySQL versions 5.1, 5.5, and 5.6.
AWS region	Aurora DB clusters can only be created in the US East (N. Virginia) (us-east-1), US West (Oregon) (us-west-2), or EU (Ireland) (eu-west-1) regions.	Available in all AWS regions.

Feature	Amazon RDS for Aurora	Amazon RDS for MySQL
MySQL storage engine	Supports only InnoDB. Tables from other storage engines are automatically converted to InnoDB. For information on converting existing MySQL tables to InnoDB and importing into an Aurora cluster, see Migrating Data to an Amazon Aurora DB Cluster (p. 415). Because Amazon Aurora only supports the InnoDB engine, the `NO_ENGINE_SUBSTITUTION` option of the `SQL_MODE` database parameter is enabled. This disables the ability to create an in-memory table, unless that table is specified as `TEMPORARY`.	Supports both MyISAM and InnoDB.
Read Replicas with a different storage engine than the master instance	MySQL (non-RDS) Read Replicas that replicate with an Aurora DB cluster can only use InnoDB.	Read Replicas can use both MyISAM and InnoDB.
Database engine parameters	Some parameters apply to the entire Aurora DB cluster and are managed by DB cluster parameter groups. Other parameters apply to each individual DB instance in a DB cluster and are managed by DB parameter groups. For more information, see Appendix: DB Cluster and DB Instance Parameters (p. 437).	Parameters apply to each individual DB instance or Read Replica and are managed by DB parameter groups.

Creating an Amazon Aurora DB Cluster

An Amazon Aurora DB cluster is made up of instances that are compatible with MySQL and a cluster volume that represents data copied across three Availability Zones as a single, virtual volume. There are two types of instances in a DB cluster: a *primary instance* and *Aurora Replicas*.

The primary instance performs all of the data modifications to the DB cluster and also supports read workloads. Each DB cluster has one primary instance. An Aurora Replica supports only read workloads. Each DB instance can have up to 15 Aurora Replicas. You can connect to any instance in the DB cluster using an endpoint address.

The following topic shows how to create an Aurora DB cluster and then add an Aurora Replica for that DB cluster.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 7) section before you can create a DB cluster.

This topic describes how you can create an Amazon Aurora DB cluster using the AWS Management Console. For simple instructions on connecting to your Aurora DB cluster, see Connecting to an Amazon Aurora DB Cluster (p. 413). For a detailed guide on connecting to an Amazon Aurora DB cluster, go to RDS Aurora Connectivity.

DB Cluster Prerequisites

The following are prerequisites to create a DB cluster.

VPC

An Amazon Aurora DB cluster can only be created in an Amazon Virtual Private Cloud (VPC) with at least two subnets in at least two Availability Zones.

If you are using the Amazon RDS console to create your Aurora DB cluster, then you can have Amazon RDS automatically create a VPC for you. Alternatively, you can use an existing VPC or create a new VPC for your Aurora DB cluster. Your VPC must have at least two subnets in order for you to use it with an Amazon Aurora DB cluster. For more infromation, see How to Create a VPC for Use with Amazon Aurora (p. 407). For information on VPCs, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).

> **Note**
> You can communicate with an EC2 instance that is not in a VPC and an Amazon Aurora DB cluster using ClassicLink. For more information, see Using ClassicLink to Connect an EC2-Classic Instance to a DB Instance in an Amazon VPC (p. 569).

If you don't have a default VPC or you have not created a VPC, you can have Amazon RDS automatically create a VPC for you when you create an Aurora DB cluster using the RDS console. Otherwise, you must do the following:

- Create a VPC with at least two subnets in at least two Availability Zones.
- Specify a VPC security group that authorizes connections to your Aurora DB cluster. For information, go to Working with a DB Instance in a VPC (p. 564).
- Specify an RDS DB subnet group that defines at least two subnets in the VPC that can be used by the Aurora DB cluster. For information, see the Working with DB Subnet Groups section in Using Amazon RDS with Amazon Virtual Private Cloud (VPC) (p. 562).

Additional Prerequisites

- If you are connecting to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS operations. For more information, see Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources (p. 88).

  If you are using an IAM account to access the Amazon Aurora console, you must first log on to the AWS Management Console with your IAM account, and then go to the Aurora console at https://console.aws.amazon.com/rds.
- If you want to tailor the configuration parameters for your DB cluster, you must specify a DB parameter group with the required parameter settings. For information about creating or modifying a DB parameter group, see Working with DB Parameter Groups (p. 523).
- You must determine the TCP/IP port number you will specify for your DB cluster. The firewalls at some companies block connections to the default Aurora port (3306). If your company firewall blocks the default port, choose another port for your DB cluster. All instances in a DB cluster use the same port.

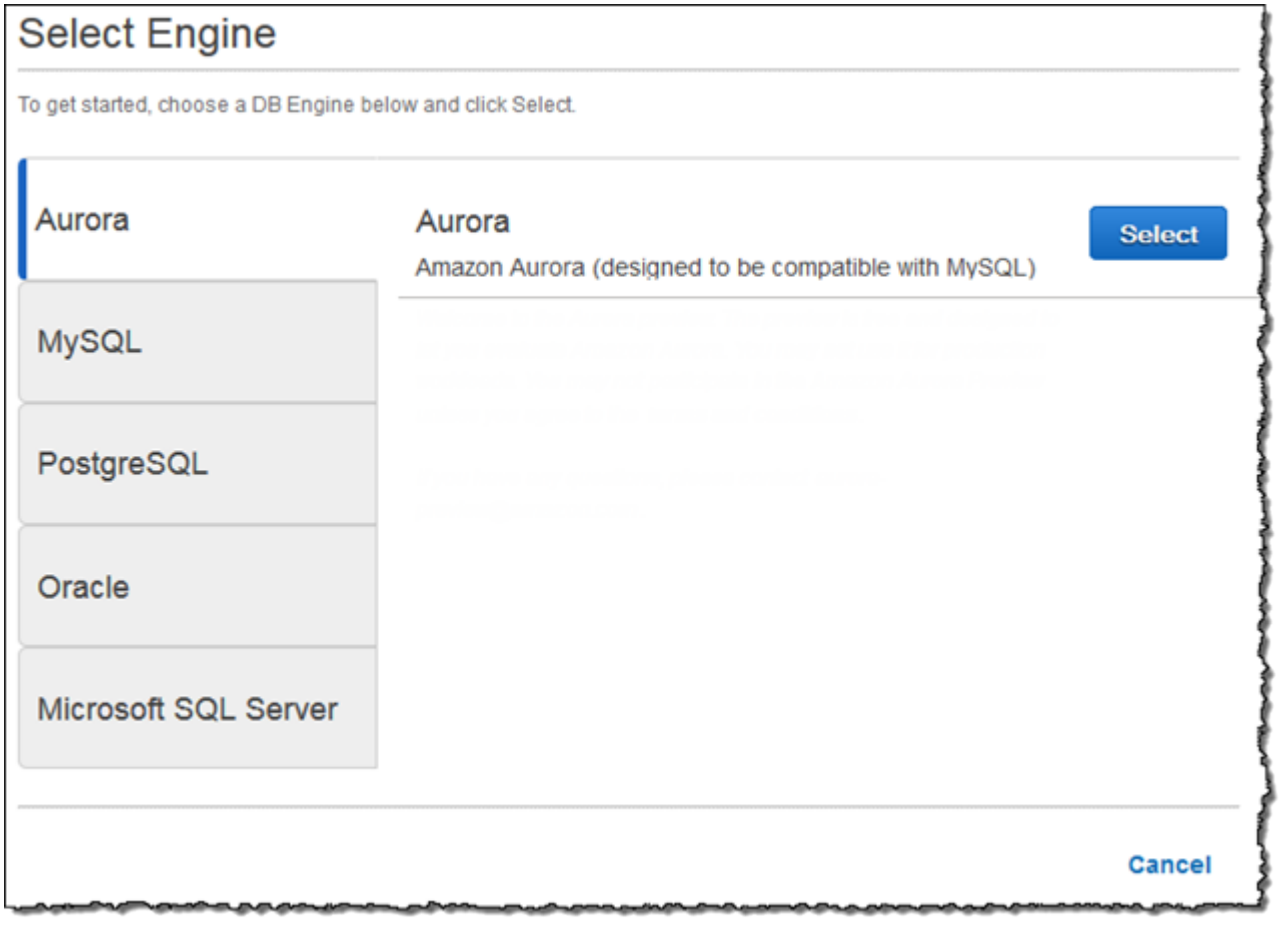
Using the AWS Management Console to Launch an Aurora DB Cluster and Create an Aurora Replica

Launching an Aurora DB Cluster

The following procedures describe how to use the AWS Management Console to launch an Aurora DB cluster and create an Aurora Replica.

To launch an Aurora DB cluster using the console

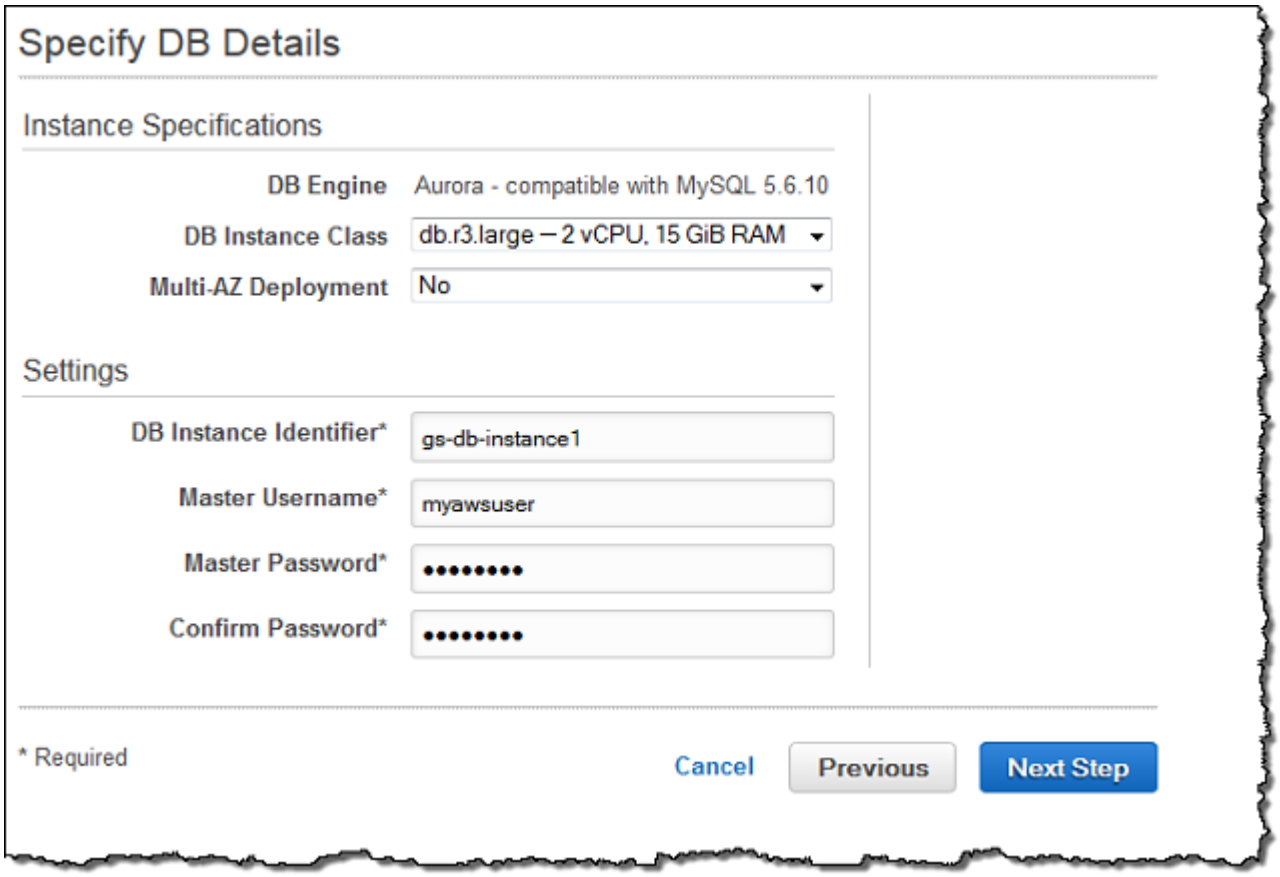
1. Open the Amazon RDS for Aurora console at https://console.aws.amazon.com/rds.
2. In the top-right corner of the AWS Management Console, select the region in which you want to create the DB cluster.
3. In the left navigation pane, click **DB Instances**.
4. Click **Launch DB Instance** to start the Launch DB Instance wizard. The wizard opens on the **Select Engine** page.
5. On the Select Engine page, select whether you want to have Amazon RDS create a new VPC and DB subnet group for you, or use an existing VPC, and then click the **Select** button for the Aurora DB engine. For information on creating a VPC for use with an Aurora DB cluster, see How to Create a VPC for Use with Amazon Aurora (p. 407).



6. On the **Specify DB Details** page, specify your DB cluster information. The following table shows settings for a DB instance.

For This Option...	Do this
DB Instance Class	Select a DB instance class that defines the processing and memory requirements for each instance in the DB cluster. Aurora supports the `db.r3.large`, `db.r3.xlarge`, `db.r3.2xlarge`, `db.r3.4xlarge`, and `db.r3.8xlarge` DB instance classes. For more information about DB instance class options, see DB Instance Class (p. 65).
Multi-AZ Deployment	Determine if you want to create Aurora Replicas in other Availability Zones for failover support. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 69).
DB Instance Identifier	Type a name for the primary instance in your DB cluster. This identifier will be used in the endpoint address for the primary instance of your DB cluster. The DB instance identifier has the following constraints: • It must contain from 1 to 63 alphanumeric characters or hyphens. • Its first character must be a letter. • It cannot end with a hyphen or contain two consecutive hyphens. • It must be unique for all DB instances per AWS account, per region.
Master Username	Type a name using alphanumeric characters that you will use as the master user name to log on to your DB cluster. The default privileges granted to the master user name account include: `create, drop, references, event, alter, delete, index, insert, select, update, create temporary tables, lock tables, trigger, create view, show view, alter routine, create routine, execute, create user, process, show databases, grant option`.
Master Password	Type a password that contains from 8 to 41 printable ASCII characters (excluding /,", and @) for your master user password.

A typical **Specify DB Details** page looks like the following.

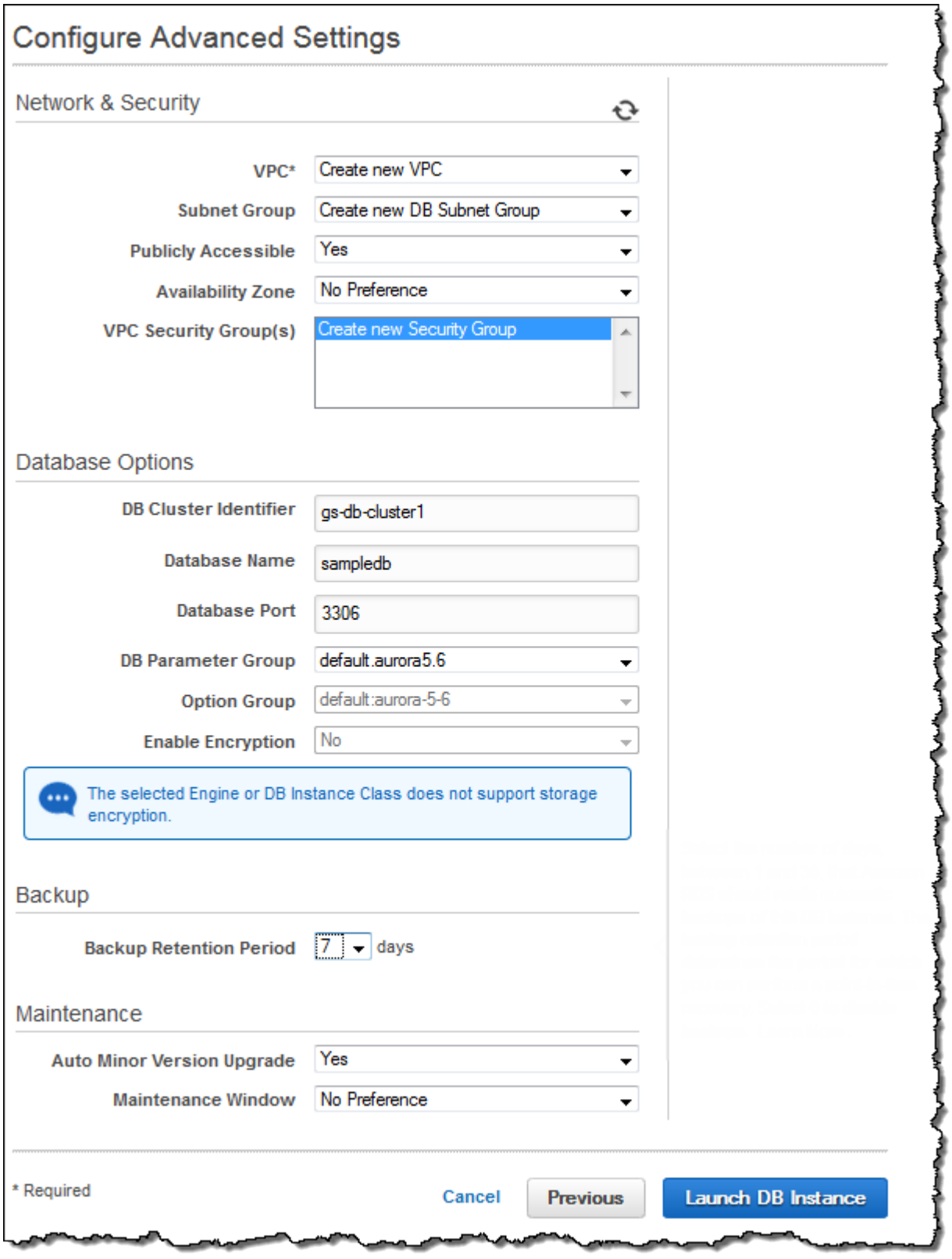


7. Confirm your master password and click **Next**.
8. On the **Configure Advanced Settings** page, you can customize additional settings for your Aurora DB cluster. The following table shows the advanced settings for a DB cluster.

For This Option...	Do This
VPC	Select the VPC that will host the DB cluster. Select **Create a New VPC** to have Amazon RDS create a VPC for you. For more information, see DB Cluster Prerequisites (p. 399) earlier in this topic.
Subnet Group	Select the DB subnet group to use for the DB cluster. Select **Create a New DB Subnet Group** to have Amazon RDS create a DB subnet group for you. For more information, see DB Cluster Prerequisites (p. 399) earlier in this topic.
Publicly Accessible	Select `Yes` to give the DB cluster a public IP address; otherwise, select `No`. The instances in your DB cluster can be a mix of both public and private DB instances. For more information about hiding instances from public access, see Hiding a DB Instance in a VPC from the Internet (p. 565).
Availability Zone	Determine if you want to specify a particular Availability Zone. For more information about Availability Zones, see Regions and Availability Zones (p. 69).

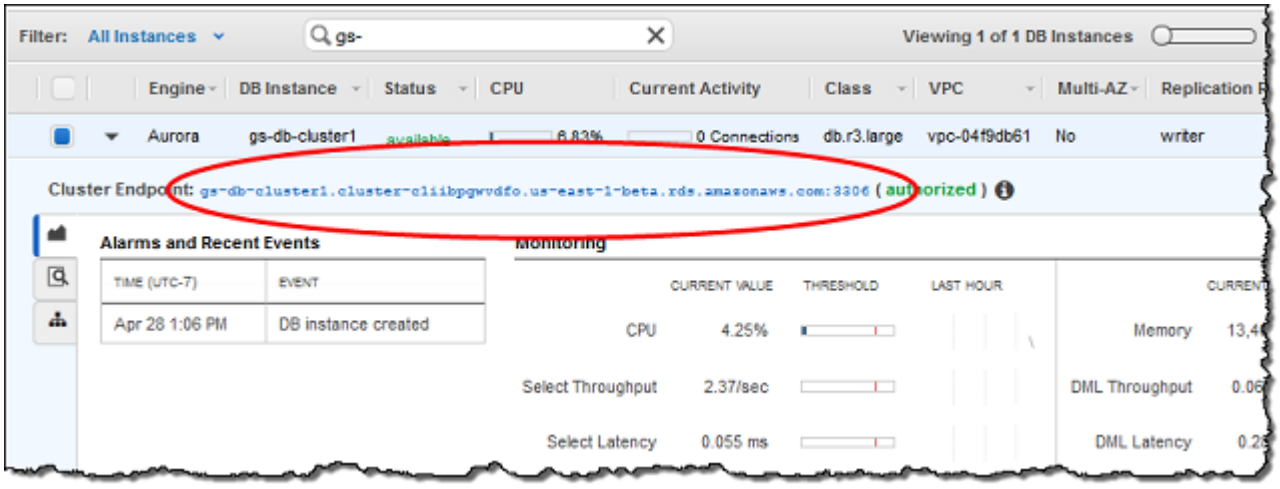
For This Option...	Do This
VPC Security Group(s)	Select one or more VPC security groups to secure network access to the DB cluster. Select **Create a New VPC Security Group** to have Amazon RDS create a VPC security group for you. For more information, see DB Cluster Prerequisites (p. 399) earlier in this topic.
DB Cluster Identifier	Type a name for your DB cluster that is unique for your account in the region you selected. This identifier will be used in the endpoint addresses for the Aurora Replicas in your DB cluster. The DB cluster identifier has the following constraints: • It must contain from 1 to 63 alphanumeric characters or hyphens. • Its first character must be a letter. • It cannot end with a hyphen or contain two consecutive hyphens. • It must be unique for all DB instances per AWS account, per region. .
Database Name	Type a name for your database of up to 8 alpha-numeric characters. If you don't provide a name, Amazon RDS will not create a database on the DB cluster you are creating.
Database Port	Specify the port that applications and utilities will use to access the database. Aurora DB clusters default to the default MySQL port, 3306. The firewalls at some companies block connections to the default MySQL port. If your company firewall blocks the default port, choose another port for the new DB cluster.
Parameter Group	Select a parameter group. Aurora has a default parameter group you can use, or you can create your own parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 523).
Option Group	Select an option group. Aurora has a default option group you can use, or you can create your own option group. For more information about option groups, see Working with Option Groups (p. 510).
Backup Retention Period	Select the length of time, from 1 to 35 days, that Aurora will retain backup copies of the database. Backup copies can be used for point-in-time restores (PITR) of your database down to the second.
Auto Minor Version Upgrade	Select `Yes` if you want to enable your Aurora DB cluster to receive minor MySQL DB Engine version upgrades automatically when they become available.
Maintenance Window	Select the weekly time range during which system maintenance can occur.

A typical **Configure Advanced Settings** page looks like the following.



9. Click **Launch DB Instance** to launch your Aurora DB instance, and then click **Close** to close the wizard.

On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to available, you can connect to the primary instance for your DB cluster. Depending on the DB instance class and store allocated, it can take several minutes for the new instance to be available.



Note the port and the endpoint of the primary instance. Use the endpoint and port of the primary instance in your JDBC and ODBC connection strings for any application that performs write or read operations.

Creating an Aurora Replica Using the Console

After creating the primary instance for your Aurora DB cluster, you can add up to 15 Aurora Replicas by using the Create Aurora Replica wizard.

> **Note**
> Amazon Aurora also supports replication with an external MySQL database, or an RDS MySQL DB instance. When using Amazon Aurora, your RDS MySQL DB instance must be in the same region. For more information, see Replication with MySQL (p. 423).

To create an Aurora Replica by using the AWS Management Console

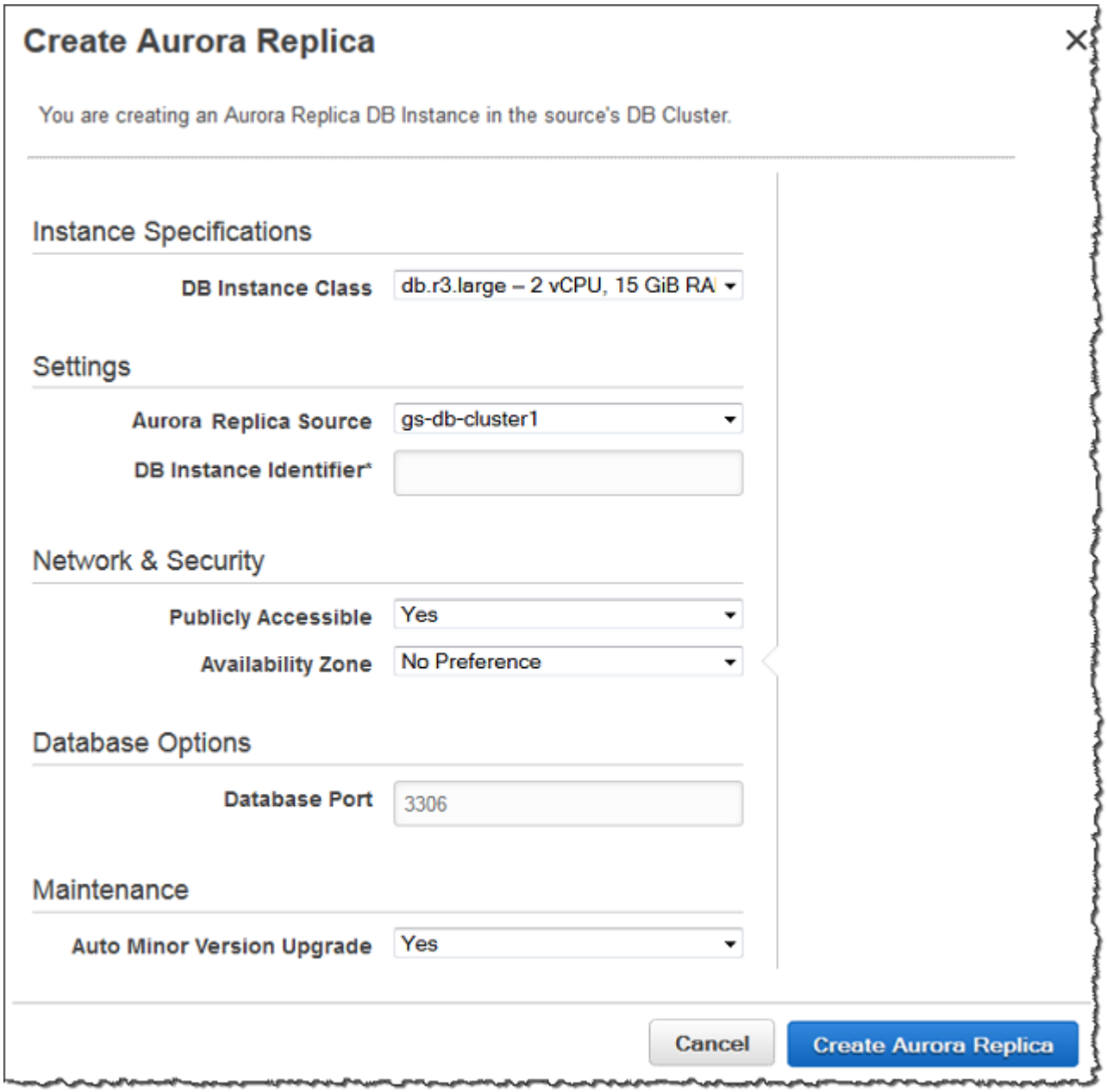
This example creates an Aurora Replica in the US East (N. Virginia) region.

1. Open the Amazon RDS for Aurora console at https://console.aws.amazon.com/rds.
2. In the left navigation pane, click **Instances**.
3. Click to select the check box to the left of the primary instance for your Aurora DB cluster.
4. Click **Instance Actions**, and then click **Create Aurora Replica**.
5. On the Create Aurora Replica page, specify options for your Aurora Replica. The following table shows settings for an Aurora Replica.

For This Option...	Do This
DB Instance Class	Select a DB instance class that defines the processing and memory requirements for the Aurora Replica. Aurora supports the `db.r3.large`, `db.r3.xlarge`, `db.r3.2xlarge`, `db.r3.4xlarge`, and `db.r3.8xlarge` DB instance classes. For more information about DB instance class options, see DB Instance Class (p. 65).

For This Option...	Do This
Aurora Replica Source	Select the identifier of the primary instance to create an Aurora Replica for.
DB Instance Identifier	Type a name for the instance that is unique for your account in the region you selected. You might choose to add some intelligence to the name such as including the region and DB engine you selected, for example **`aurora-read-instance1`**.
Publicly Accessible	Select `Yes` to give the Aurora Replica a public IP address; otherwise, select `No`. For more information about hiding Aurora Replicas from public access, see Hiding a DB Instance in a VPC from the Internet (p. 565).
Availability Zone	Determine if you want to specify a particular Availability Zone. The list includes only those Availability Zones that are mapped by the DB subnet group you specified earlier. For more information about Availability Zones, see Regions and Availability Zones (p. 69).
Database Port	The port for an Aurora Replica is the same as the port for the DB cluster.
Auto Minor Version Upgrade	Select `Yes` if you want to enable your Aurora Replica to receive minor Aurora DB engine version upgrades automatically when they become available.

A typical **Create Aurora Replica** page looks like the following.



6. Click **Create Aurora Replica** to create the Aurora Replica.

Note the endpoint of the Aurora Replica. Use the endpoint of the Aurora Replica in your JDBC and ODBC connection strings for any application that performs only read operations.

How to Create a VPC for Use with Amazon Aurora

The following sections discuss how to create a VPC for use with Amazon Aurora.

> **Note**
> For a helpful and detailed guide on connecting to an Amazon Aurora DB cluster, you can go to RDS Aurora Connectivity.

Create a VPC and Subnets

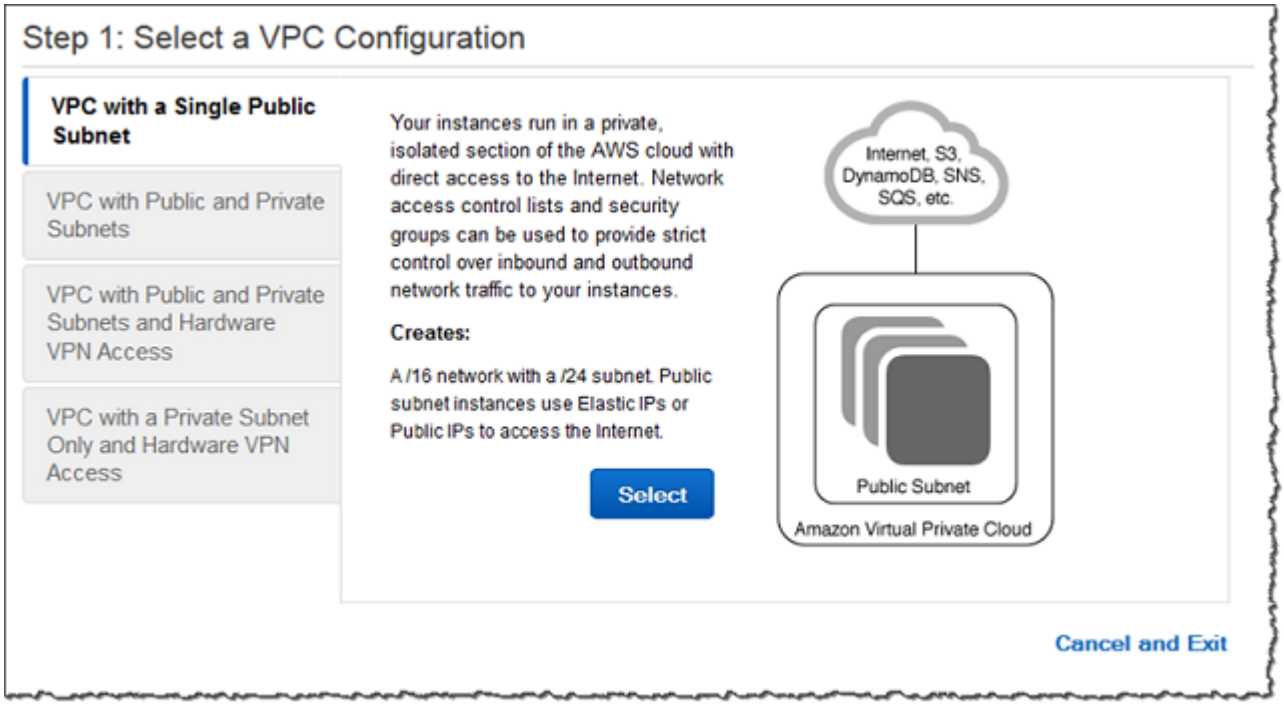
You can only create an Amazon Aurora DB cluster in an Amazon Virtual Private Cloud (VPC) with at least two subnets in at least two Availability Zones. You can create an Aurora DB cluster in the default VPC for your AWS account, or you can create a user-defined VPC. For information, see Amazon RDS and Amazon Virtual Private Cloud (VPC) (p. 73).

Amazon RDS will, optionally, create a VPC and subnet group for you to use with your Amazon Aurora DB cluster. This can be helpful if you have never created a VPC, or if you would like to create a new VPC that is separate from your other VPCs. If you want Amazon RDS to create a VPC and subnet group for you, then skip this procedure and go to Create a DB Cluster (p. 47).

> **Note**
> All VPC and EC2 resources that you use with your Aurora DB cluster must be in the US East (N. Virginia), US West (Oregon), or EU (Ireland) regions.

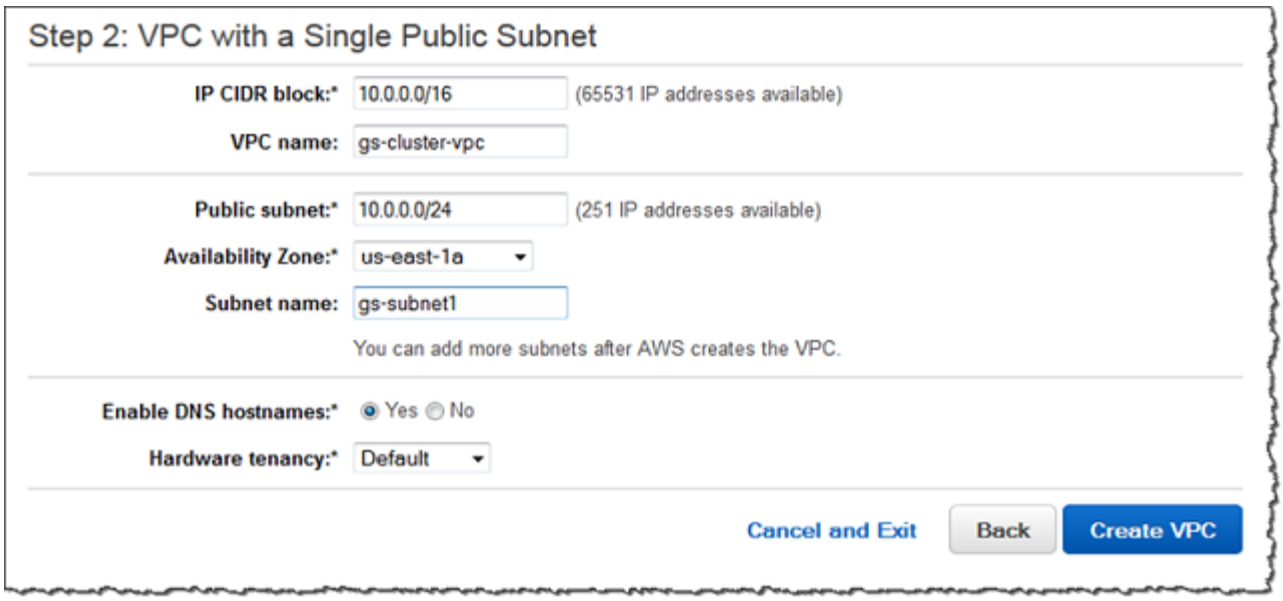
To create a VPC for use with an Aurora DB cluster

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.amazonaws.cn/vpc/.
2. In the top-right corner of the AWS Management Console, select the region to create your VPC in. This example uses the US East (N. Virginia) region. Aurora is only supported for the US East (N. Virginia), US West (Oregon), or EU (Ireland) regions.
3. In the upper-left corner, click **VPC Dashboard**. Click **Start VPC Wizard** to begin creating a VPC.
4. In the Create VPC wizard, click **VPC with a Single Public Subnet**. Click **Select**.


Step 1: Select a VPC Configuration
VPC with a Single Public Subnet
VPC with Public and Private Subnets
VPC with Public and Private Subnets and Hardware VPN Access
VPC with a Private Subnet Only and Hardware VPN Access
Your instances run in a private, isolated section of the AWS cloud with direct access to the Internet. Network access control lists and security groups can be used to provide strict control over inbound and outbound network traffic to your instances.
Creates:
A /16 network with a /24 subnet. Public subnet instances use Elastic IPs or Public IPs to access the Internet.
Select
Internet, S3, DynamoDB, SNS, SQS, etc.
Public Subnet
Amazon Virtual Private Cloud
Cancel and Exit

5. Set the following values in the **Create VPC** panel:
   - **IP CIDR block:** `10.0.0.0/16`
   - **VPC name:** `gs-cluster-vpc`
   - **Public subnet:** `10.0.0.0/24`
   - **Availability Zone:** `us-east-1a`

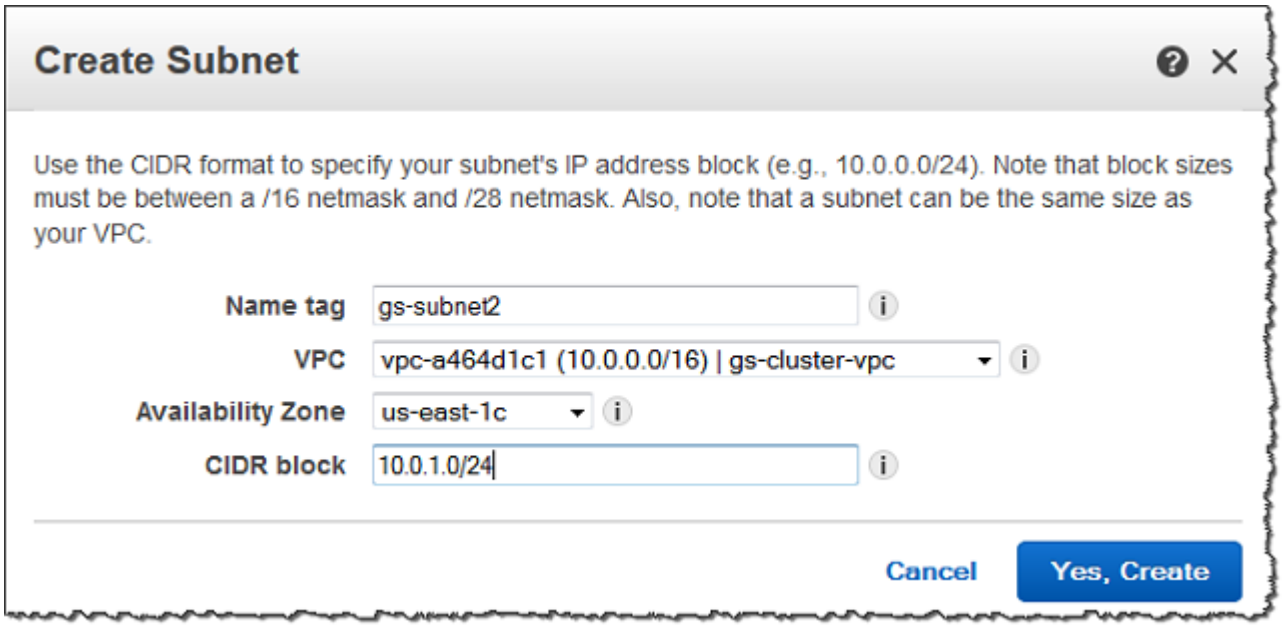
- **Subnet name:** `gs-subnet1`
- **Enable DNS hostnames:** `Yes`
- **Hardware tenancy:** `Default`



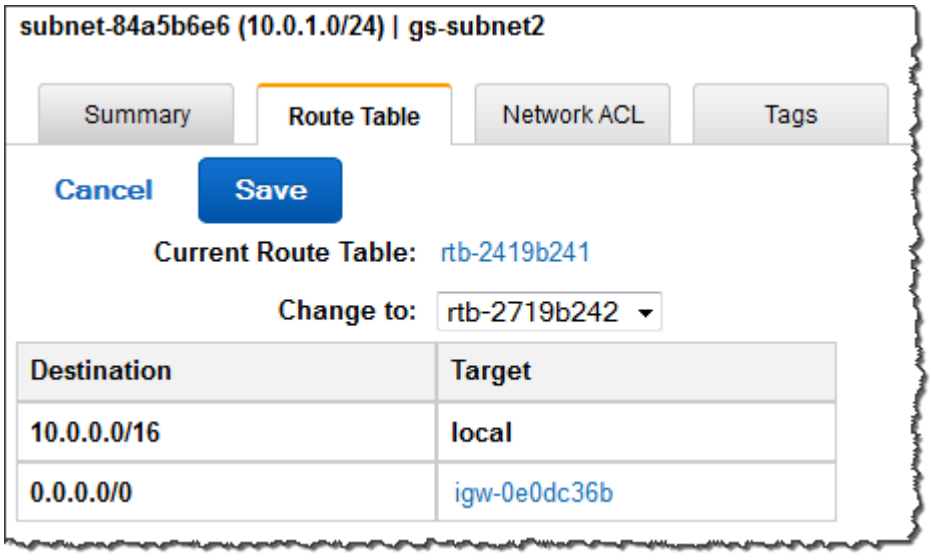
6. Click **Create VPC**.
7. When your VPC has been created, click **Close** on the notification page.

To create additional subnets

1. To add the second to your VPC, in the VPC Dashboard click **Subnets**, and then click **Create Subnet**. An Amazon Aurora DB cluster requires at least two VPC subnets.
2. Set the following values in the **Create Subnet** panel:
   - **Name tag:** `gs-subnet2`
   - **VPC:** Select the VPC that you created in the previous step, for example: `vpc-a464d1c1 (10.0.0.0/16) | gs-cluster-vpc`.
   - **Availability Zone:** `us-east-1c`
   - **CIDR block:** `10.0.1.0/24`


Create Subnet
Use the CIDR format to specify your subnet's IP address block (e.g., 10.0.0.0/24). Note that block sizes must be between a /16 netmask and /28 netmask. Also, note that a subnet can be the same size as your VPC.
Name tag
gs-subnet2
VPC
vpc-a464d1c1 (10.0.0.0/16) | gs-cluster-vpc
Availability Zone
us-east-1c
CIDR block
10.0.1.0/24
Cancel
Yes, Create

3. Click **Yes Create**.
4. To ensure that the second subnet that you created uses the same route table as the first subnet, in the VPC Dashboard, click **Subnets**, and then select the first subnet that was created for the VPC, `gs-subnet1`. Click the **Route Table** tab, and note the **Current Route Table**, for example: `rtb-2719b242`.
5. In the list of subnets, select the second subnet, `gs-subnet2`. Select the **Route Table** tab, and then click **Edit**. In the **Change to** list, select the route table from the previous step, for example: `rtb-2719b242`. Click **Save** to save your selection.



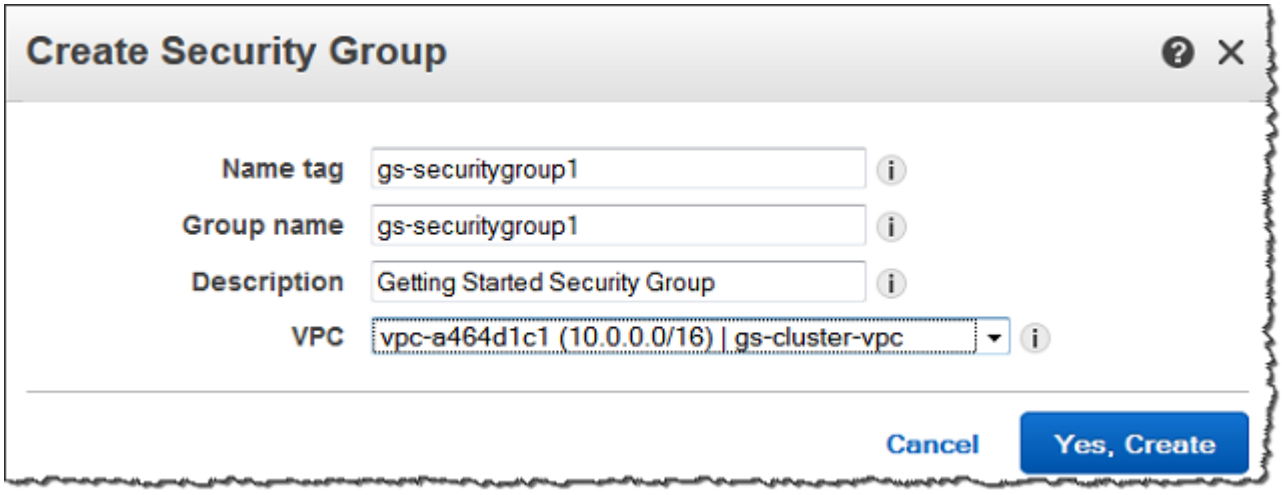
Create a Security Group and Add Inbound Rules

After you've created your VPC and subnets, the next step is to create a security group and add inbound rules.

To create a security group

The last step in creating a VPC for use with your Amazon Aurora DB cluster is to create a VPC security group, which will identify which network addresses and protocols are allowed to access instances in your VPC.

1. In the VPC Dashboard, click **Security Groups**, and then click **Create Security Group**.
2. Set the following values in the **Create Security Group** panel:
   - **Name tag:** `gs-securitygroup1`
   - **Group name:** `gs-securitygroup1`
   - **Description:** `Getting Started Security Group`
   - **VPC:** Select the VPC that you created earlier, for example: `vpc-a464d1c1 (10.0.0.0/16) | gs-cluster-vpc`.



3. Click **Yes, Create** to create the security group.

To add inbound rules to the security group

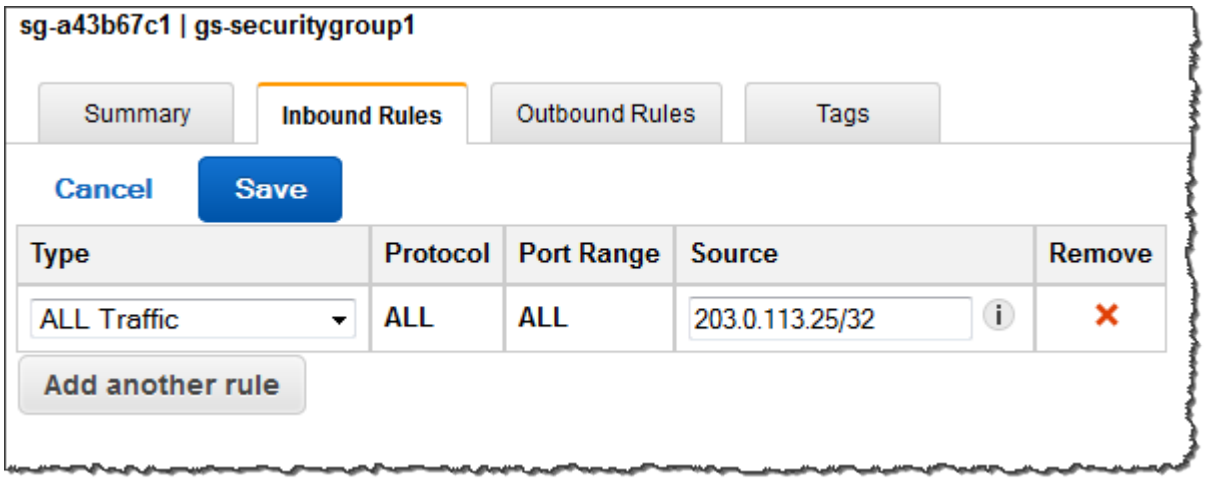
To connect to your Aurora DB instance, you will need to add an inbound rule to your VPC security group that allows inbound traffic to connect.

1. Determine the IP address that you will be using to connect to the Aurora cluster. You can use the service at http://checkip.amazonaws.com to determine your public IP address. If you are connecting through an ISP or from behind your firewall without a static IP address, you need to find out the range of IP addresses used by client computers.

   Caution
   If you use `0.0.0.0/0`, you enable all IP addresses to access your DB cluster. This is acceptable for a short time in a test environment, but it's unsafe for production environments. In production, you'll authorize only a specific IP address or range of addresses to access your DB cluster.

2. In the VPC Dashboard, click **Security Groups**, and then select the `gs-securitygroup1` security group that you created in the previous procedure.

3. Select the **Inbound Rules** tab, and then click the **Edit** button.
4. Set the following values for your new inbound rule:
   - **Type:** `All Traffic`
   - **Source:** The IP address or range from the previous step, for example `203.0.113.25/32`.



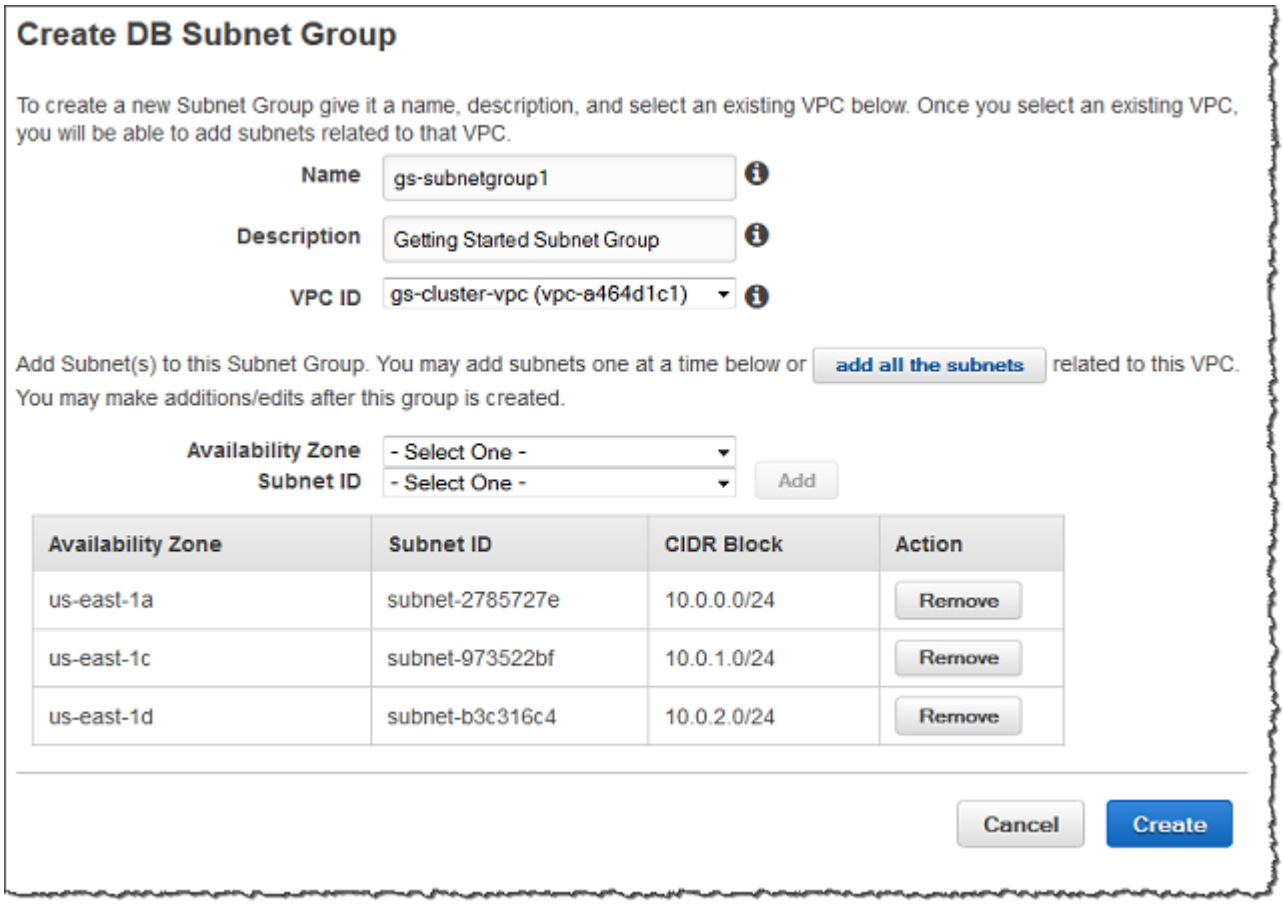
5. Click **Save** to save your settings.

Create an RDS Subnet Group

The last thing that you need before you can create an Aurora DB cluster is a DB subnet group. Your RDS DB subnet group identifies the subnets that your DB cluster will use from the VPC that you created in the previous steps. Your DB subnet group must include at least two subnets in at least two Availability Zones.

To create a DB subnet group for use with your Aurora DB cluster

1. Open the Amazon RDS for Aurora console at https://console.aws.amazon.com/rds.
2. Select **Subnet Groups**, and then click **Create DB Subnet Group**.
3. Set the following values for your new DB subnet group:
   - **Name:** `gs-subnetgroup1`
   - **Description:** `Getting Started Subnet Group`
   - **VPC ID:** Select the VPC that you created in the previous procedure, for example, `vpc-a464d1c1`.
4. Click **add all the subnets** to add the subnets for the VPC that you created in earlier steps. You can also add each subnet individually by selecting the **Availability Zone** and the **Subnet ID** and clicking **Add**.



5. Click **Yes, Create** to create the subnet group.

Connecting to an Amazon Aurora DB Cluster

You can connect to an Aurora DB instance using the same tools that you use to connect to a MySQL database, including using the same public key for Secure Sockets Layer (SSL) connections. You can use the endpoint and port information from the primary instance or Aurora Replicas in your Amazon Aurora DB cluster in the connection string of any script, utility, or application that connects to a MySQL DB instance. In the connection string, specify the DNS address from the primary instance or Aurora Replica endpoint as the host parameter, and specify the port number from the endpoint as the port parameter.

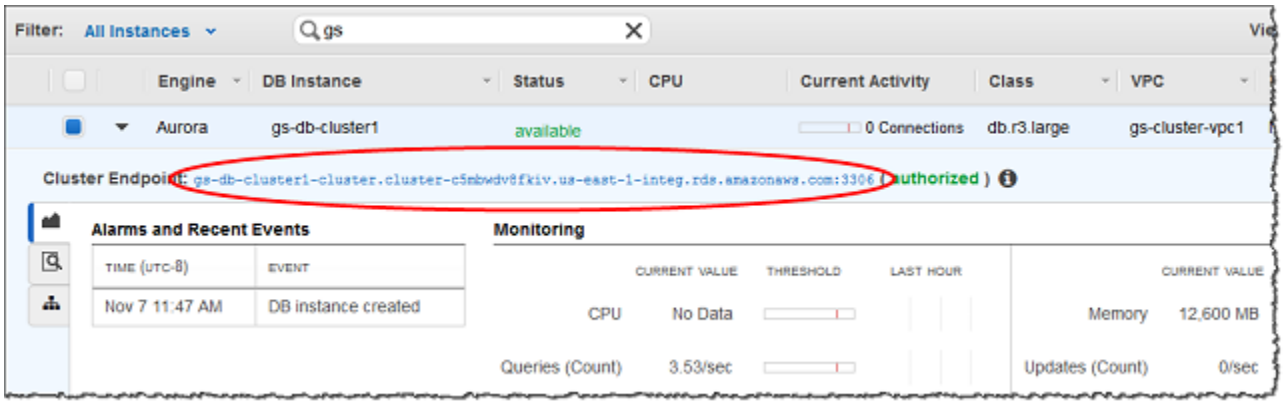
Once you have a connection to your Amazon Aurora DB cluster, you can execute any SQL command that is compatible with MySQL version 5.6. For more information about MySQL 5.6 SQL syntax, go to the MySQL 5.6 Reference Manual.

> **Note**
> For a helpful and detailed guide on connecting to an Amazon Aurora DB cluster, you can go to RDS Aurora Connectivity.

In the details view for your DB cluster you will find the cluster endpoint, which you can use in your MySQL connection string. The endpoint is made up of the domain name and port for your DB cluster. For example, if an endpoint value is `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com:3306`, then you specify the following values in a MySQL connection string:

- For host or host name, specify `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- For port, specify `3306`

The cluster endpoint connects you to the primary instance for the DB cluster. You can perform both read and write operations using the cluster endpoint. Your DB cluster can also have up to 15 Aurora Replicas that support read-only access to the data in your DB cluster. The primary instance and each Aurora Replica each have a unique endpoint that is independent of the cluster endpoint and allows you to connect to a specific DB instance in the cluster directly. The cluster endpoint will always point to the primary instance. If the primary instance fails and is replaced, then the cluster endpoint will point to the new primary instance.



You can connect to an Amazon Aurora DB cluster by using tools like the MySQL command line utility. For more information on using the MySQL utility, go to mysql - The MySQL Command Line Tool in the MySQL documentation. A GUI-based application you can use to connect is MySQL Workbench. For more information, go to the Download MySQL Workbench page.

You can use SSL encryption on connections to an Amazon Aurora DB instance. For information, see Using SSL with a MySQL DB Instance (p. 123).

Connecting with SSL

To connect using SSL, use the MySQL utility as described in the following procedure.

To connect to a DB cluster with SSL using the MySQL utility

1. Download the public key for the Amazon RDS signing certificate from https://rds.amazonaws.com/doc/rds-ssl-ca-cert.pem. Note that this will download a file named `rds-ssl-ca-cert.pem`.
2. Type the following command at a command prompt to connect to a DB cluster with SSL using the MySQL utility. For the -h parameter, substitute the endpoint DNS name for your instance. For the --ssl_ca parameter, substitute the SSL certificate file name as appropriate. Type the master user password when prompted.

```
PROMPT> mysql -h mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com
--ssl-ca=rds-ssl-ca-cert.pem --ssl-verify-server-cert
```

You will see output similar to the following:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

```
Your MySQL connection id is 350
Server version: 5.1.32-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Because an Amazon Aurora DB cluster can only be created in an Amazon Virtual Private Cloud (VPC), connections to an Amazon Aurora DB cluster from AWS instances that are not in a VPC have been required to use the public endpoint address of the Amazon Aurora DB cluster. However, you can now communicate with an EC2 instance that is not in a VPC and an Amazon Aurora DB cluster using ClassicLink. For more information, see Using ClassicLink to Connect an EC2-Classic Instance to a DB Instance in an Amazon VPC (p. 569).

For general instructions on constructing Amazon RDS MySQL connection strings and finding the public key for SSL connections, see Connecting to a DB Instance Running the MySQL Database Engine (p. 137).

Troubleshooting Aurora Connection Failures

Note
For a helpful and detailed guide on connecting to an Amazon Aurora DB cluster, you can go to RDS Aurora Connectivity.

Common causes of connection failures to a new Aurora DB cluster are as follows:

- The DB cluster was created using a VPC that does not allow connections from your device. To fix this failure, modify the VPC to allow connections from your device, or create a new VPC for your DB cluster that allows connections from your device. For an example, see Create a VPC and Subnets (p. 408).
- The DB cluster was created using the default port of 3306, and your company has firewall rules blocking connections to that port from devices in your company network. To fix this failure, recreate the instance with a different port.

Migrating Data to an Amazon Aurora DB Cluster

You can migrate data to an Amazon Aurora DB cluster either from an Amazon RDS snapshot or from a MySQL instance running externally to Amazon RDS, as described following.

Note
Because Amazon Aurora is compatible with MySQL, you can migrate data from your MySQL database by setting up replication between your MySQL database, and an Amazon Aurora DB cluster. We recommend that your MySQL database run MySQL version 5.5 or later. For more information, see Amazon Aurora Replication (p. 394).

Migrating an RDS MySQL Snapshot to Aurora

You can migrate a DB snapshot of an Amazon RDS MySQL DB instance to create an Aurora DB cluster. The new DB cluster will be populated with the data from the original Amazon RDS MySQL DB instance. The DB snapshot must have been made from an Amazon RDS DB instance running MySQL 5.6.

You can migrate either a manual or automated DB snapshot. After the DB cluster is created, you can then create optional Aurora Replicas.

The general steps you must take are as follows:

1. Determine the amount of space to provision for your Amazon Aurora DB cluster. For more information, see How Much Space Do I Need? (p. 416)
2. Use the console to create the snapshot in the region where the Amazon RDS MySQL 5.6 instance is located. For information about creating a DB snapshot, see Creating a DB Snapshot.
3. If the DB snapshot is not in the region as your DB cluster, use the Amazon RDS console to copy the DB snapshot to that region. For information about copying a DB snapshot, go to Copying a DB Snapshot.
4. Use the console to migrate the DB snapshot and create an Amazon Aurora DB cluster with the same databases as the original DB instance of MySQL 5.6.

Caution
Amazon RDS limits each AWS account to one snapshot copy into each region at a time.

How Much Space Do I Need?

When you migrate a snapshot of a MySQL DB instance into an Aurora DB cluster, Aurora uses an EBS volume to format the data from the snapshot before migrating it. There are some cases where additional space is needed to format the data for migration. When migrating data into your DB cluster, observe the following guidelines and limitations:

- Although Amazon Aurora supports up to 64 TB of storage, the process of migrating a snapshot into an Aurora DB cluster is limited by the size of the EBS volume of the snapshot, and therefore is limited to a maximum size of 6 TB.
- Tables that are not MyISAM tables can be up to 6 TB in size. However, if you have MyISAM tables then Aurora must use additional space in the volume to convert the tables to be Aurora compatible. If this is the case, then ensure that none of these tables being migrated from your MySQL DB instance exceeds 3 TB in size.

Reducing the Amount of Space Required to Migrate Data into Amazon Aurora

You might want to modify your database schema prior to migrating it into Amazon Aurora. This can be helpful in the following cases:

- You want to speed up the migration process.
- You are unsure of how much space you need to provision.
- You have attempted to migrate your data and the migration has failed due to a lack of provisioned space.

You can make the following changes to improve the process of migrating a database into Amazon Aurora.

Important
Be sure to perform these updates on a new DB instance restored from a snapshot of a production database, rather than on a production instance. You can then migrate the data from the snapshot of your new DB instance into your Amazon Aurora DB cluster to avoid any service interruptions on your production database.

Table Type	Limitation/Guideline
MyISAM Tables	Amazon Aurora supports InnoDB tables only. If you have MyISAM tables in your database, then those tables must be converted before being migrated into Amazon Aurora. The conversion process requires additional space for the MyISAM to InnoDB conversion during the migration procedure. To reduce your chances of running out of space or to speed up the migration process, convert all of your MyISAM tables to InnoDB tables before migrating them. The size of the resulting InnoDB table is equivalent to the size required by Amazon Aurora for that table. To convert a MyISAM table to InnoDB, run the following command: `alter table <schema>.<table_name> engine=innodb, algorithm=copy;`

You can use the following SQL script on your existing MySQL DB instance to list the tables in your database that are MyISAM tables.

```
-- This script examines a MySQL database for conditions that will block
-- migrating the database into Amazon's Aurora DB.
-- It needs to be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.

-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
    as major_minor
  ) as T
where major_minor <> 506;


-- List MyISAM tables and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM Tables.`,
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
```

```
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  );
```

The script produces output as shown in the following example. The example shows two tables that must be converted from MyISAM to InnoDB. The output also includes the approximate size of each table in MB.

```
+--------------------+------------------+
| ==> MyISAM Tables. | Approx size (MB) |
+--------------------+------------------+
| test.name_table    |          2102.25 |
| test.my_table      |            65.25 |
+--------------------+------------------+
2 rows in set (0.01 sec)
```

Migrating a DB Snapshot by Using the Console

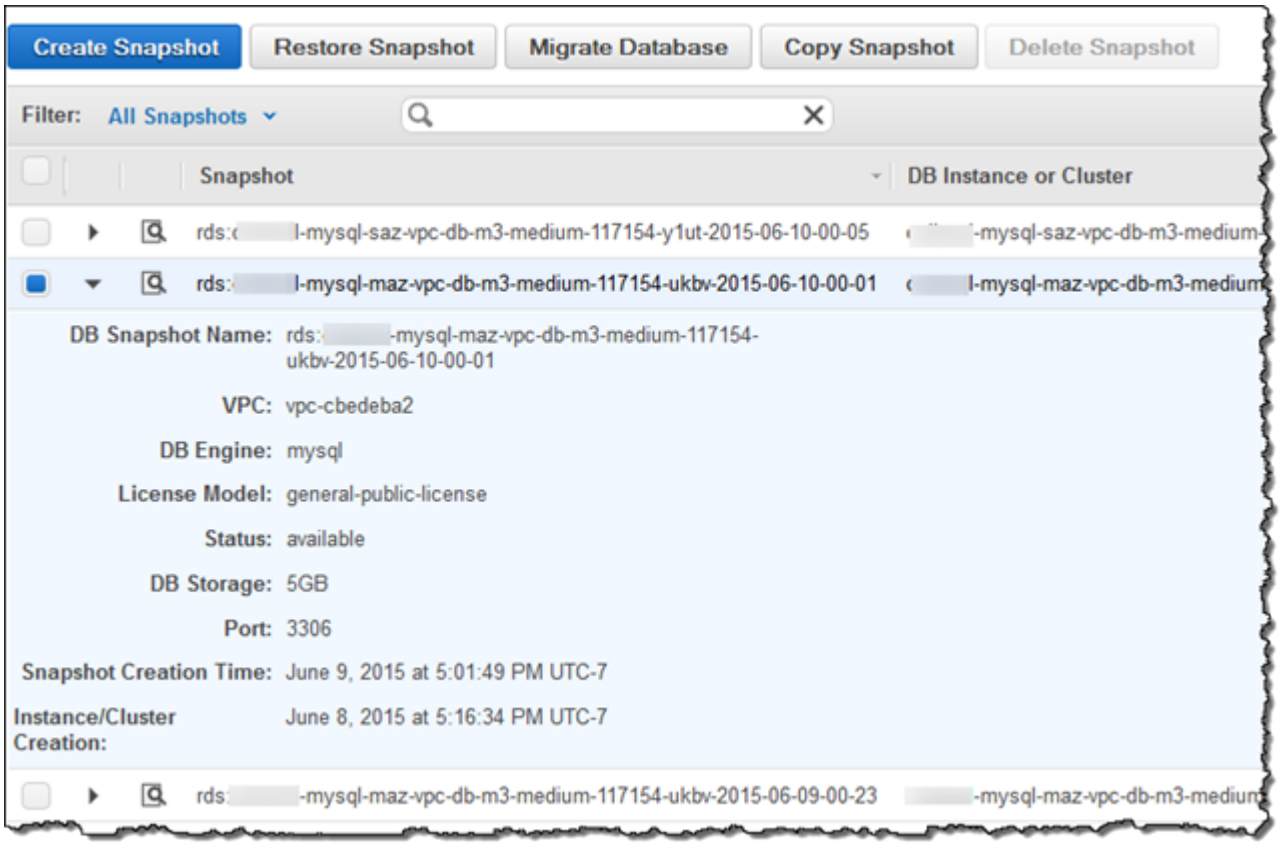
You can migrate a DB snapshot of an Amazon RDS MySQL DB instance to create an Aurora DB cluster. The new DB cluster will be populated with the data from the original Amazon RDS MySQL DB instance. The DB snapshot must have been made from an Amazon RDS DB instance running MySQL 5.6. For information about creating a DB snapshot, see Creating a DB Snapshot.

If the DB snapshot is not in the region where you want to locate your data, use the Amazon RDS console to copy the DB snapshot to that region. For information about copying a DB snapshot, see Copying a DB Snapshot.

When you migrate the DB snapshot by using the console, the console takes the actions necessary to create both the DB cluster and the primary instance.

To migrate a MySQL 5.6 DB snapshot by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Choose **Snapshots**.
3. On the **Snapshots** page, choose the snapshot that you want to migrate into an Aurora DB cluster.
4. Choose **Migrate Database**.



5. Set the following values on the **Migrate Database** page:
   - **DB Instance Class:** Select a DB instance class that has the required storage and capacity for your database, for example `db.r3.large`. Aurora cluster volumes automatically grow as the amount of data in your database increases, up to a maximum size of 64 terabytes (TB). So you only need to select a DB instance class that meets your current storage requirements. For more information, see Amazon Aurora Storage (p. 394).
   - **DB Instance Identifier:** Type a name for the DB cluster that is unique for your account in the region you selected. This identifier will be used in the endpoint addresses for the instances in your DB cluster. You might choose to add some intelligence to the name such as including the region and DB Engine you selected, for example **`aurora-cluster1`**.

     The DB instance identifier has the following constraints:
     - It must contain from 1 to 63 alphanumeric characters or hyphens.
     - Its first character must be a letter.
     - It cannot end with a hyphen or contain two consecutive hyphens.
     - It must be unique for all DB instances per AWS account, per region.
   - **VPC:** If you have an existing VPC, then you can use that VPC with your Amazon Aurora DB cluster by selecting your VPC identifier, for example, `vpc-a464d1c1`. For information on using an existing VPC, see How to Create a VPC for Use with Amazon Aurora (p. 407).

     Otherwise, you can choose to have Amazon RDS create a VPC for you by selecting **Create a new VPC**.
   - **Subnet Group:** If you have an existing subnet group, then you can use that subnet group with your Amazon Aurora DB cluster by selecting your subnet group identifier, for example, `gs-subnet-group1`.

Otherwise, you can choose to have Amazon RDS create a subnet group for you by selecting **Create a new subnet group**.

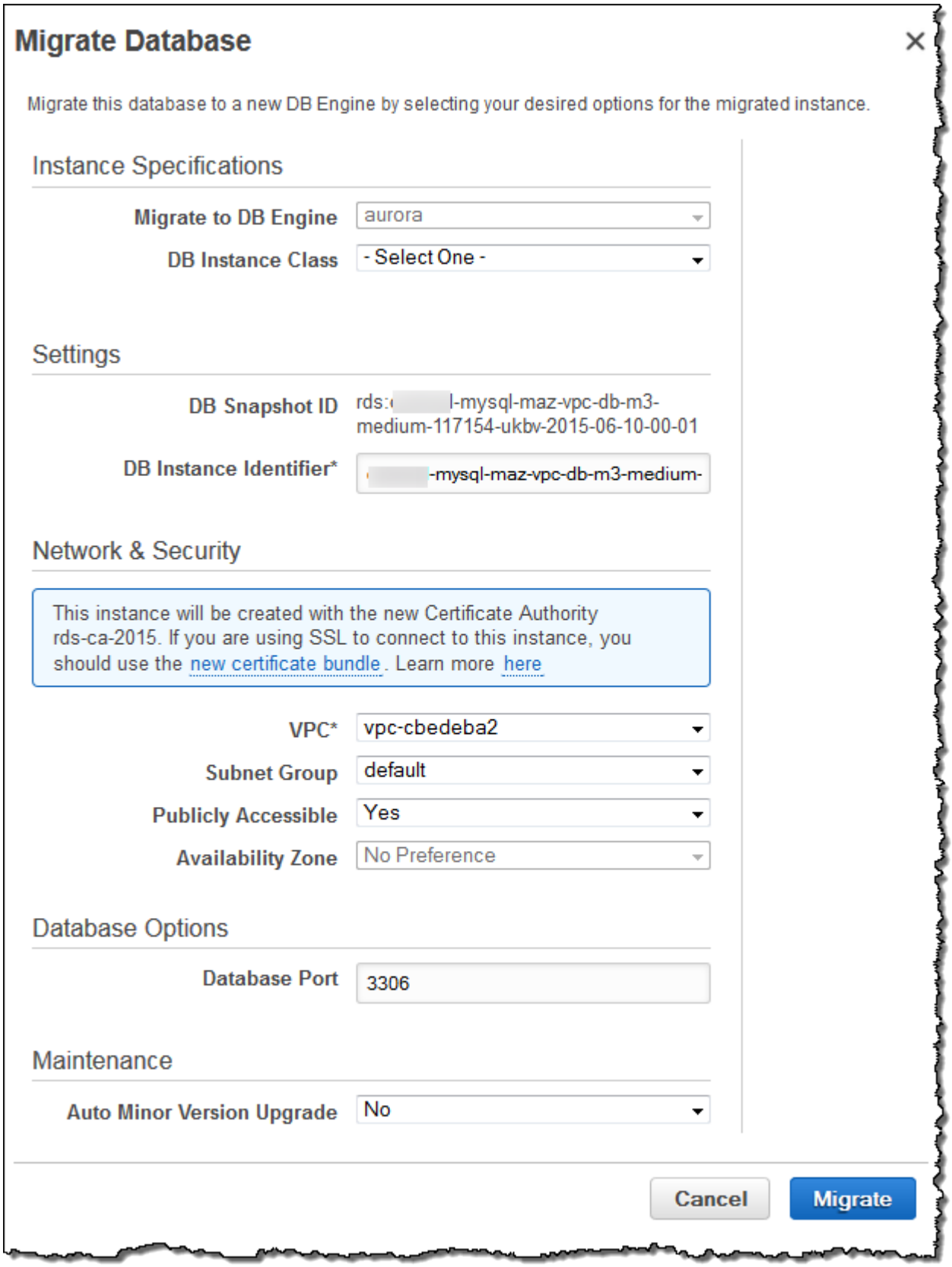
- **Publicly Accessible:** Select `No` to specify that instances in your DB cluster can only be accessed by resources inside of your VPC. Select `Yes` to specify that instances in your DB cluster can be accessed by resources on the public network. The default is `Yes`.

  Note
  Your production DB cluster might not need to be in a public subnet, because only your application servers will require access to your DB cluster. If your DB cluster doesn't need to be in a public subnet, set **Publicly Accessible** to `No`.

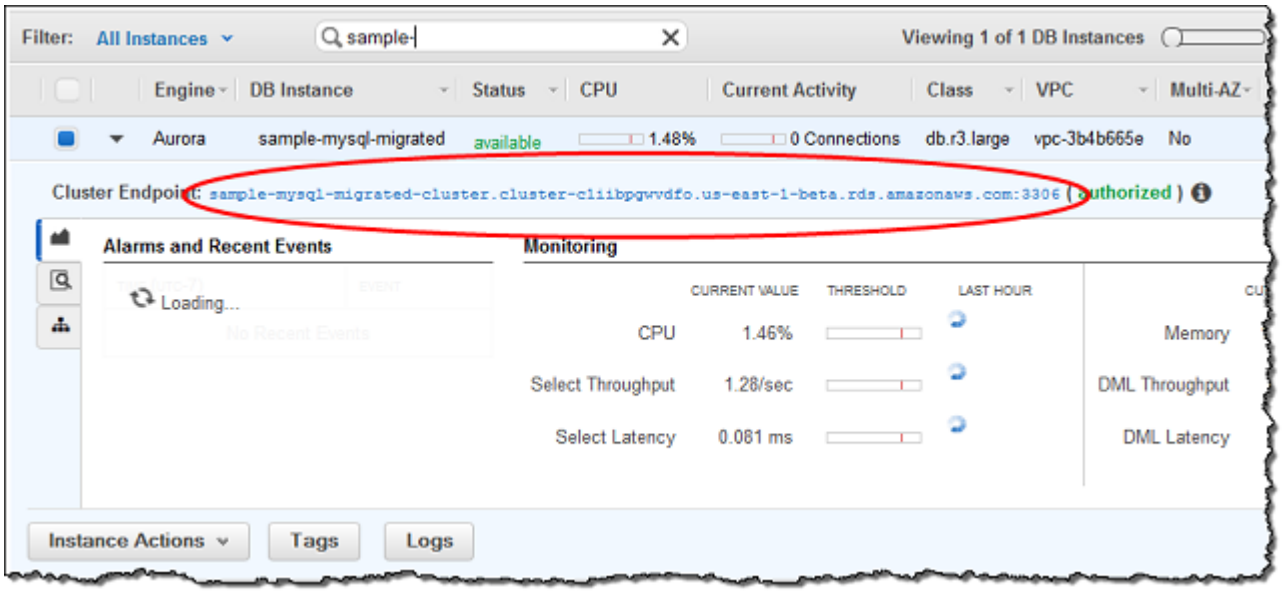
- **Availability Zone:** Select the Availability Zone to host the primary instance for your Aurora DB cluster. To have Amazon RDS select an Availability Zone for you, select `No Preference`.
- **Database Port:** Type the default port that will be used when connecting to instances in the DB cluster. The default is `3306`.

  Note
  You might be behind a corporate firewall that does not allow access to default ports such as the MySQL default port, 3306. In this case, provide a port value that your corporate firewall allows. Remember that port value later when you connect to the Aurora DB cluster.

- **Auto Minor Version Upgrade:** Select `Yes` if you want to enable your Aurora DB cluster to receive minor MySQL DB Engine version upgrades automatically when they become available.


Migrate Database
Migrate this database to a new DB Engine by selecting your desired options for the migrated instance.
Instance Specifications
Migrate to DB Engine
aurora
DB Instance Class
- Select One -
Settings
DB Snapshot ID
rds: l-mysql-maz-vpc-db-m3-medium-117154-ukbv-2015-06-10-00-01
DB Instance Identifier*
-mysql-maz-vpc-db-m3-medium-
Network & Security
This instance will be created with the new Certificate Authority rds-ca-2015. If you are using SSL to connect to this instance, you should use the new certificate bundle. Learn more here
VPC*
vpc-cbedeba2
Subnet Group
default
Publicly Accessible
Yes
Availability Zone
No Preference
Database Options
Database Port
3306
Maintenance
Auto Minor Version Upgrade
No
Cancel
Migrate

6. Click the **Migrate** to migrate your DB snapshot.
7. Select **Instances** and click the arrow icon to show the DB cluster details and monitor the progress of the migration. On the details page, you will find the cluster endpoint used to connect to the primary instance of the DB cluster. For more information on connecting to an Amazon Aurora DB cluster, see Connecting to an Amazon Aurora DB Cluster (p. 413).



Migrating from MySQL to Amazon Aurora with Reduced Downtime

When importing data from a MySQL database that supports a live application to an Amazon Aurora DB cluster, you can use the procedure documented in Importing Data to an Amazon RDS MySQL DB Instance with Reduced Downtime (p. 147) to reduce the amount of time that service to your data is interrupted in order to migrate your data to Aurora. The procedure can especially help if you are working with a very large database, because you can reduce the cost of the import by minimizing the amount of data that is passed across the network to AWS.

The procedure lists steps to transfer a copy of your database data to an Amazon EC2 instance and import the data into a new Amazon RDS MySQL DB instance. Because Amazon Aurora is compatible with MySQL, you can instead use an Amazon Aurora DB cluster for the target Amazon RDS MySQL DB instance.

Replication with Amazon Aurora

Read scaling is accomplished in an Aurora DB cluster using Aurora Replicas. Aurora Replicas are independent endpoints in an Aurora DB cluster. Up to 15 Aurora Replicas can be distributed across the Availability Zones that the DB cluster spans within a single region. The cluster volume is made up of multiple copies of the data for the DB cluster, but the data in the cluster volume is represented as a single, logical volume to the primary and Aurora Replicas in the DB cluster. As a result, all Aurora Replicas return the same data for query results with minimal replica lag—usually well less than 100 milliseconds after the primary instance has written an update. Replica lag will vary depending on the database change rate. That is, during periods where a large amount of writes are occurring to the database, you might see an increase in replica lag.

Aurora Replicas are best used for scaling read operations and increasing availability. Aurora Replicas are fully dedicated to read operations on your cluster volume (unless the Aurora Replica is promoted to the primary instance). Write operations are managed by the primary instance. Because the cluster volume is shared among all instances in your DB cluster, no additional work is required to replicate a copy of the data for each Aurora Replica. In contrast, MySQL Read Replicas must replay, on a single thread, all write operations from the master DB instance to their local data store, which can affect MySQL Read Replica's ability to support large volumes of read traffic.

Aurora Replicas are failover targets. That is, if the primary instance fails, then an Aurora Replica is promoted to the primary instance with only a brief interruption during which read and write requests made to the primary instance will fail with an exception. If your Aurora DB cluster does not include any Aurora Replicas, then the primary instance is recreated during a failure event. However, promoting an Aurora Replica is much faster than recreating the primary instance. For high-availability scenarios, we recommend that you create one or more Aurora Replicas, of the same DB instance class as the primary instance, in different Availability Zones for your Aurora DB cluster. For more information on Aurora Replicas as failover targets, see Fault Tolerance for an Aurora DB Cluster (p. 430).

For details on how to create an Aurora Replica, see Creating an Aurora Replica Using the Console (p. 405).

Monitoring Aurora Replication

You can monitor how far an Aurora Replica is lagging behind the primary instance of your Aurora DB cluster by monitoring the Amazon CloudWatch **Replica Lag** metric. Because Aurora Replicas read from the same cluster volume as the primary instance, the **Replica Lag** metric has a different meaning for an Aurora DB cluster. The **Replica Lag** metric for an Aurora Replica indicates the lag for the page cache of the Aurora replica compared to that of the primary instance.

If you need the most current value for Aurora Replica lag, you can query the `mysql.ro_replica_status` table in your Aurora DB cluster and check the value in the `Replica_lag_in_msec` column. This column value is provided to Amazon CloudWatch as the value for the **Replica Lag** metric. The values in the `mysql.ro_replica_status` are also provided in the `INFORMATION_SCHEMA.REPLICA_HOST_STATUS` table in your Aurora DB cluster.

For more information on monitoring RDS instances and CloudWatch metrics, see Monitoring Amazon RDS (p. 571).

Replication with MySQL

Because Amazon Aurora is compatible with MySQL, you can set up replication between a MySQL database, and an Amazon Aurora DB cluster. We recommend that your MySQL database run MySQL version 5.5 or later.

Replication Between Amazon Aurora and an External MySQL Database

> **Important**
> There is a known issue when Amazon Aurora DB cluster is the replication slave where replication will pause unexpectedly. In this case, the CloudWatch **Replica Lag** will continue to grow. If this occurs, you must restore the Amazon Aurora DB cluster from the most recent snapshot and set up replication with the restored DB cluster as the new replication slave.

To set up replication between an Amazon Aurora DB cluster as the replica and a MySQL database that you manage outside of Amazon RDS, follow the instructions in Replication with a MySQL Instance Running External to Amazon RDS (p. 162), with the following requirements and recommendations:

- The `sql_mode` option for your MySQL DB instance must be set to `0`, or must not be included in your MySQL `my.cnf` file.

- Monitor failover events for the Amazon Aurora DB cluster that is your replica. If a failover occurs, then the DB cluster that is your replica might be recreated on a new host with a different network address. For information on how to monitor failover events, see Using Amazon RDS Event Notification (p. 575).
- Maintain the binlogs on your master instance until you have verified that they have been applied to the replica. This maintenance ensures that you can restore your master instance in the event of a failure.

Replication Between Amazon Aurora and an RDS MySQL DB Instance

Important
Your Amazon Aurora DB cluster and an RDS MySQL DB instance must be in the same region to replicate; Amazon Aurora supports the US East (N. Virginia), US West (Oregon), and EU (Ireland) regions.

To set up replication between an Amazon Aurora DB cluster as the replica, and an RDS MySQL DB instance, follow these steps (shown for the US East (N. Virginia) region):

1. Create a Read Replica of your RDS MySQL DB instance. For information on creating a Read Replica, see Creating a Read Replica (p. 476). Do not delete this Read Replica until you no longer are replicating between your Amazon Aurora DB cluster and your RDS MySQL DB instance.
2. On the newly created Read Replica, stop replication by calling the mysql.rds_stop_replication (p. 181) command (the `mysql.rds_stop_replication` command is only available for MySQL versions 5.5.33 and later and 5.6.13 and later.
3. Create a snapshot of the Read Replica while it is **Stopped**. Import the snapshot into your Amazon Aurora DB cluster using the instructions in Migrating Data to an Amazon Aurora DB Cluster (p. 415).
4. Take a manual snapshot of the Aurora DB cluster that will be the replication slave prior to setting up replication. If you need to reestablish replication with the DB cluster as a replication slave, you can restore the Aurora DB cluster from this snapshot instead of having to import the data from your MySQL DB instance into a new Aurora DB cluster.
5. On the Read Replica, run the `SHOW SLAVE STATUS` command and retrieve the current binary log file name from the `Master_Log_File` field and the log file position from the `Read_Master_Log_Pos` field.
6. Connect to the Aurora DB cluster and issue the mysql.rds_set_external_master (p. 178) command to start replication with your MySQL DB instance using the binary log file name and location, for example:

```
CALL mysql.rds_set_external_master (mydbinstance.123456789012.us-east-
1.rds.amazonaws.com', 3306,
    'repl_user', '<password>', 'mysql-bin-changelog.000031', 107, 0);
CALL mysql.rds_start_replication;
```

7. While your Aurora DB cluster and your RDS MySQL DB instance are replicating, monitor the amount of storage available for your MySQL RDS DB instance. The binary logs used for replication are not automatically deleted with this configuration and will use up additional storage space.

   If you need to delete the binary logs on your MySQL RDS DB instance to free up storage space, first ensure that there is no replication lag on your Amazon Aurora DB cluster. You can do this by running the `SHOW SLAVE STATUS` command on your Aurora DB cluster and checking the **Seconds behind master** field. If the **Seconds behind master** field is `0`, then there is no replica lag.

   When there is no replica lag, you can delete the binary log files by calling the mysql.rds_start_replication (p. 180) command on the Read Replica that you created in step 1. When replication is started between your RDS MySQL DB instance and the Read Replica, RDS will delete the binary logs once they have been replayed on the Read Replica. After you have deleted enough

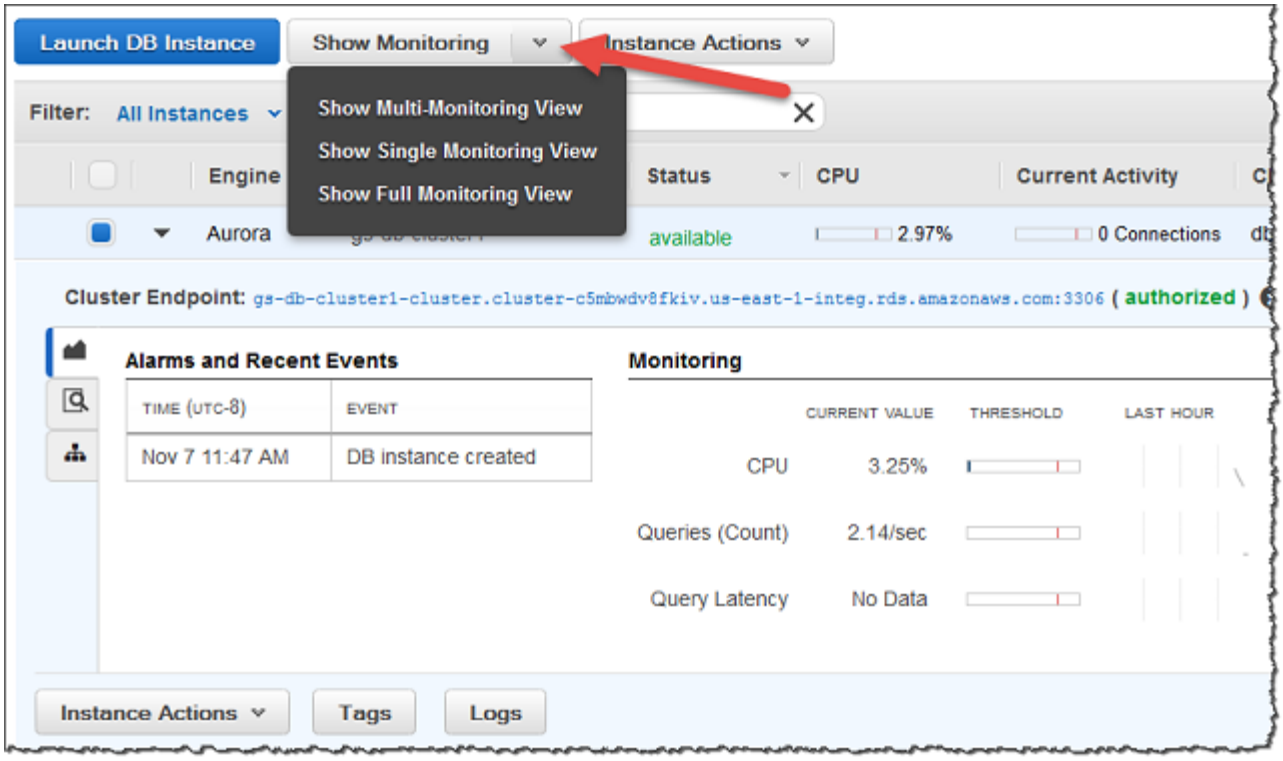
binary logs to free up the amount of storage space that you need, then stop replication by calling the mysql.rds_stop_replication (p. 181) command on the Read Replica that you created in step 1. This will ensure that the binary logs on your RDS MySQL DB instance are not deleted before they are replayed on the Amazon Aurora DB cluster.

Monitoring an Amazon Aurora DB Cluster

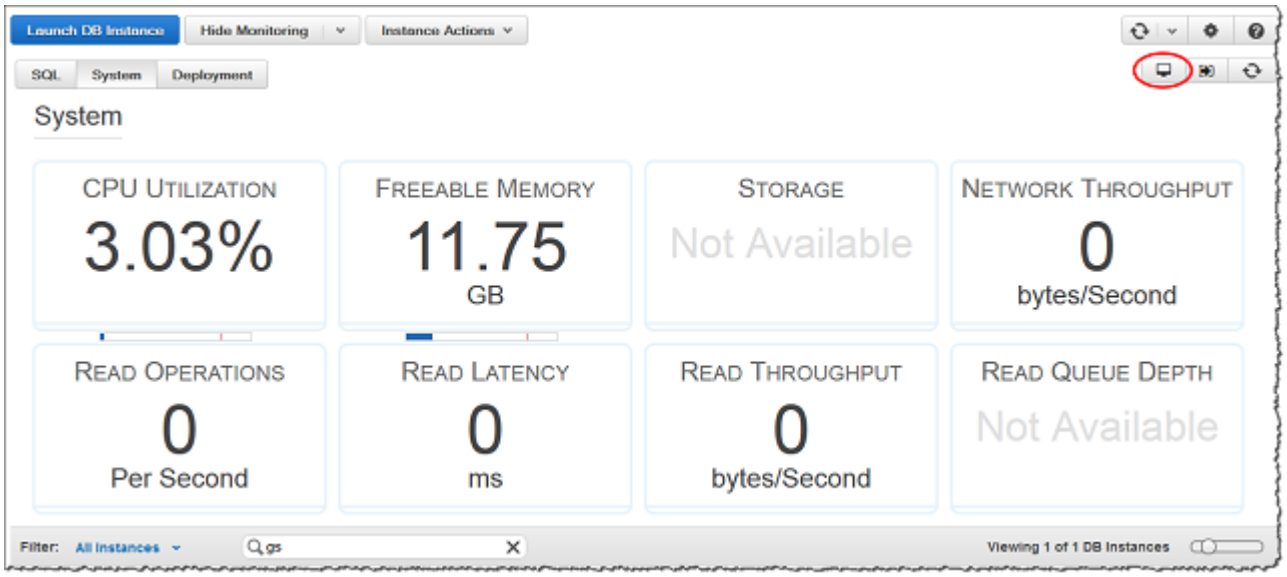
You can view details about your DB cluster by using the Amazon RDS console. The Amazon Aurora console provides a number of metrics for you to monitor the health and performance of your Aurora DB cluster. For a detailed list, see Aurora Metrics (p. 426).

To view the details of a DB cluster using the Amazon RDS console

1. Open the Amazon RDS for Aurora console at https://console.aws.amazon.com/rds.
2. In the left navigation pane, click **Instances**.
3. Click the check box to the left of the DB cluster you need information about. Then click the **Show Monitoring** drop-down menu. Select the option for how you want to view your Aurora metrics. The Aurora console provides three options for viewing metrics.
   - **Show Multi-Monitoring View**–Shows a summary of Aurora metrics. Each metric includes a graph showing the metric monitored over a specific time span.
   - **Show Single Monitoring View**–Shows a single metric at a time with more detail. Each metric includes a graph showing the metric monitored over a specific time span.
   - **Show Full Monitoring View**–Shows a summary of Aurora metrics without graphs. **Full Monitoring View** includes an option for full-screen viewing.



4. If you selected **Full Monitoring View**, you can click the full screen button to view only your metrics in full-screen mode.



For more information and other options for monitoring RDS instances, see Monitoring Amazon RDS (p. 571).

Aurora Metrics

Aurora provides several metrics that you can monitor to determine the health of your DB cluster. You can view these metrics in the RDS console. The following tables describe the metrics available for instances in Aurora DB clusters.

Aurora System Monitoring

Metric	Description
CPU Utilization	The percentage of CPU used by a DB instance.
Freeable memory	The amount of available random access memory, in gigabytes (GB).
Storage	The amount of storage used by the DB cluster, in GB. This value affects the cost of the Aurora DB cluster. For pricing information, go to the Amazon RDS product page.
Network receive throughput	The amount of network throughput received from clients by each instance in the DB cluster, in megabytes per second (mbps). This does not include network traffic between instances in the DB cluster and the cluster volume.
Network transmit throughput	The amount of network throughput sent to clients by each instance in the DB cluster, in megabytes per second (mbps). This does not include network traffic between instances in the DB cluster and the cluster volume.
Read operations	The average number of read disk I/O requests from the instance to the cluster volume per second.
Billed read operations	The average number of read disk I/O operations from the cluster volume per second.
Read latency	The elapsed time between the submission of a read I/O request and its completion, in milliseconds (ms).

Metric	Description
Read throughput	The amount of network throughput used by requests to the DB cluster, in mbps.
Read queue depth	The number of outstanding read I/O requests waiting to access the disk.
Write operations	The average number of write disk I/O requests from the instance to the cluster volume per second.
Billed write operations	The average number of write disk I/O operations to the cluster volume per second.
Write latency	The average elapsed time between the submission of a write I/O request and its completion, in ms.
Write throughput	The amount of network throughput used by responses from the DB cluster, in mbps.
Write queue depth	The number of outstanding write I/O requests waiting to access the disk.
Replica lag	For an Aurora Replica, reports the amount of lag when replicating updates from the primary instance, in ms.
Replica lag maximum	The maximum amount of lag between the primary instance and each Aurora instance in the DB cluster, in ms.
Replica lag minimum	The minimum amount of lag between the primary instance and each Aurora instance in the DB cluster, in ms.
Queue depth	The number of outstanding I/O requests (read and write) waiting to access the disk.
Free storage space	Unlike other DB engines, the Free storage space metric for Amazon Aurora reports the amount of storage available to each DB instance for temporary tables and logs, in GB. This value affects the cost of the Aurora DB cluster (or pricing information, go to the Amazon RDS product page). You can increase the amount of free storage space for an instance by choosing a larger DB instance class for your instance.
Swap usage	Not supported for Amazon Aurora.
Binary log disk usage	The amount of storage used by MySQL binary logs, in GB.
CPU credit balance	Not supported for Amazon Aurora.
CPU credit usage	Not supported for Amazon Aurora.

Aurora SQL Monitoring

Metric	Description
Select throughput	The average number of select queries per second.
DML throughput	The average number of inserts, updates, and deletes, per second.
Commit throughput	The average number of committed transactions per second.
DDL throughput	The average number of DDL requests per second.

Metric	Description
Select latency	The amount of latency for select queries, in ms.
DML latency	The amount of latency for inserts, updates, and deletes, in ms.
Commit latency	The amount of latency for committed transactions, in ms.
DDL latency	The amount of latency for DDL requests (create/alter/drop), in ms.
DB connections	The number of active connections to a DB instance.
Active transactions	The average number of current transactions executing on a DB instance per second.
Buffer cache hit ratio	The percentage of requests that are served by the Buffer cache.
Resultset cache hit ratio	The percentage of requests that are served by the Resultset cache.
Login failures	The average number of failed login attempts per second.
Blocked transactions	The average number of transactions in the database that are blocked per second.
Failed SQL statements	The average number of database commands that have failed per second.
Deadlocks	The average number of deadlocks in the database per second.

Managing an Amazon Aurora DB Cluster

The following sections discuss managing performance, scaling, fault tolerance, backup, and restoring for an Amazon Aurora DB cluster.

Managing Performance and Scaling for Aurora DB Cluster

Scaling Storage

Aurora storage automatically scales with the data in your cluster volume. As your data grows, your cluster volume storage will grow in 10 gigabyte (GB) increments up to 64 TB. The size of your cluster volume is checked on an hourly basis to determine your storage costs. For pricing information, go to the Amazon RDS product page.

Scaling Aurora DB Instances

You can scale Aurora DB instances in two ways, instance scaling and read scaling.

Instance Scaling

You can scale your DB cluster as needed by modifying the DB instance class for each DB instance in the cluster. Aurora supports several DB instance classes optimized for Aurora. The following table describes the instance class specifications.

Instance Class	vCPU	ECU	Memory (GB)
db.r3.large	2	6.5	15
db.r3.xlarge	4	13	30.5
db.r3.2xlarge	8	26	61
db.r3.4xlarge	16	52	122
db.r3.8xlarge	32	104	244

Read Scaling

You can achieve read scaling for your Aurora DB cluster by creating up to 15 Aurora Replicas in the DB cluster. Each Aurora Replica returns the same data from the cluster volume with minimal replica lag—usually considerably less than 100 milliseconds after the primary instance has written an update. As your read traffic increases, you can create additional Aurora Replicas and connect to them directly to distribute the read load for your DB cluster. Aurora Replicas don't have to be of the same DB instance class as the primary instance.

Backing Up and Restoring an Aurora DB Cluster

The following sections discuss Aurora backups and how to restore your Aurora DB cluster using the AWS Management Console.

Backups

Aurora backs up your cluster volume automatically and retains restore data for the length of the *backup retention period*. Aurora backups are continuous and incremental so you can quickly restore to any point within the backup retention period. No performance impact or interruption of database service occurs as backup data is being written. You can specify a backup retention period, from 1 to 35 days, when you create or modify a DB cluster.

If you want to retain a backup beyond the backup retention period, you can also take a snapshot of the data in your cluster volume. Storing snapshots incurs the standard storage charges for Amazon RDS. For more information about RDS storage pricing, go to Amazon Relational Database Service Pricing.

Because Aurora retains incremental restore data for the entire backup retention period, you only need to create a snapshot for data that you want to retain beyond the backup retention period. You can create a new DB cluster from the snapshot.

Restoring Data

You can recover your data by creating a new Aurora DB cluster from the backup data that Aurora retains, or from a DB cluster snapshot that you have saved. A new copy of a DB cluster created from backup data can be quickly restored to any point in time during your backup retention period. The continuous and incremental nature of Aurora backups during the backup retention period means you don't need to take frequent snapshots of your data in order to improve restore times.

To determine the latest or earliest restorable time for a DB instance, look for the `Latest Restorable Time` or `Earliest Restorable Time` values on the RDS console. The latest restorable time for a DB cluster is the most recent point at which you can restore your DB cluster, typically within 5 minutes of the current time. The earliest restorable time specifies how far back within the backup retention period that you can restore your cluster volume.

When you request a restore of your DB cluster, the new cluster volume is immediately available for both read and write operations. However, as the copy is being created, you might encounter some latency for read operations. This latency will only occur if a query requests data that has not yet been restored to the cluster volume. In that case, the data will be immediately restored to the cluster volume. Once the requested data has been restored, it will be returned to the query request.

Because your Aurora DB cluster is available for read and write operations during a restore operation, RDS shows the DB cluster status as **available** even though the restore operation has not completed. You can determine when the restore of a DB cluster is complete by checking the `Latest Restorable Time` or `Earliest Restorable Time`. The `Latest Restorable Time` and `Earliest Restorable Time` values will return NULL until the restore operation is complete. You cannot request a backup or restore operation if the `Latest Restorable Time` or `Earliest Restorable Time` values return NULL.

To restore a DB cluster to a specified time using the AWS Management Console

1. Open the Amazon RDS for Aurora console at https://console.aws.amazon.com/rds.
2. In the left navigation pane, click **Instances**. Click to select the primary instance for the DB cluster that you want to restore.
3. Click **Instance Actions**, and then click **Restore To Point In Time**.

   In the **Restore DB Cluster** window, click to select the **Use Custom Restore Time** option.
4. Type the date and time that you want to restore to in the **Use Custom Restore Time** boxes.
5. Type a name for the new, restored DB instance in the **DB Instance Identifier** box.
6. Click the **Launch DB Cluster** button to launch the restored DB cluster.

Fault Tolerance for an Aurora DB Cluster

An Aurora DB cluster is fault tolerant by design. The cluster volume spans multiple Availability Zones in a single region, and each Availability Zone contains a copy of the cluster volume data. This functionality means that your DB cluster can tolerate a failure of an Availability Zone without any loss of data and only a brief interruption of service.

In the event of a primary instance failure, Aurora will automatically failover to a new primary instance.

If the DB cluster doesn't contain any Aurora Replicas, then the primary instance will be recreated during a failure event. This failure event results in an interruption during which read and write operations will fail with an exception. Service is restored when the new primary instance is created, which typically takes less than 10 minutes. Because the Aurora cluster volume spans multiple Availability Zones, your data is preserved in the case of an instance failure or even an Availability Zone failure.

If the DB cluster has one or more Aurora Replicas, then an Aurora Replica is promoted to the primary instance during a failure event, which also results in a brief interruption. However, promoting an Aurora Replica to the primary instance is much faster than creating a new primary instance. Service is typically restored in less than 120 seconds, and often less than 60 seconds. To increase the availability of your DB cluster, we recommend that you create at one or more Aurora Replicas in different Availability Zones.

> **Note**
> Amazon Aurora also supports replication with an external MySQL database, or an RDS MySQL DB instance. Your RDS MySQL DB instance must be in the either the US East (N. Virginia), US West (Oregon), or EU (Ireland) region. For more information, see Replication with MySQL (p. 423).

Testing Amazon Aurora Using Fault Injection Queries

You can test the fault tolerance of your Amazon Aurora DB cluster by using fault injection queries. Fault injection queries are issued as SQL commands to an Amazon Aurora instance and they enable you to schedule a simulated occurrence of one of the following events:

- A crash of the master instance or an Aurora Replica
- A failure of an Aurora Replica
- A disk failure
- Disk congestion

Fault injection queries that specify a crash force a crash of the Amazon Aurora instance. The other fault injection queries result in simulations of failure events, but don't cause the event to occur. When you submit a fault injection query, you also specify an amount of time for the failure event simulation to occur for.

You can submit a fault injection query to one of your Aurora Replica instances by connecting to the endpoint for the Aurora Replica. For more information, see Aurora Endpoints (p. 393).

Testing an Instance Crash

You can force a crash of an Amazon Aurora instance using the `ALTER SYSTEM CRASH` fault injection query.

Syntax

```
ALTER SYSTEM CRASH [ INSTANCE | DISPATCHER | NODE ];
```

Options

This fault injection query takes one of the following crash types:

- **`INSTANCE`**—A crash of the MySQL-compatible database for the Amazon Aurora instance is simulated.
- **`DISPATCHER`**—A crash of the dispatcher on the master instance for the Aurora DB cluster. The *dispatcher* writes updates to the cluster volume for an Amazon Aurora DB cluster is simulated.
- **`NODE`**—A crash of both the MySQL-compatible database and the dispatcher for the Amazon Aurora instance is simulated. For this fault injection simulation, the cache is also deleted.

The default crash type is `INSTANCE`.

Testing an Aurora Replica Failure

You can simulate the failure of an Aurora Replica using the `ALTER SYSTEM SIMULATE READ REPLICA FAILURE` fault injection query.

An Aurora Replica failure will block all requests to an Aurora Replica or all Aurora Replicas in the DB cluster for a specified time interval. When the time interval completes, the affected Aurora Replicas will be automatically synced up with master instance.

Syntax

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT
```

```
READ REPLICA FAILURE [ TO ALL | TO "replica name" ]

FOR INTERVAL quantity [ YEAR | QUARTER | MONTH | WEEK| DAY | HOUR | MINUTE |
SECOND ];
```

Options

This fault injection query takes the following parameters:

- **`percentage_of_failure`—**The percentage of requests to block during the failure event. This value can be a double between 0 and 100. If you specify 0, then no requests are blocked. If you specify 100, then all requests are blocked.
- **Failure type—**The type of failure to simulate. Specify `TO ALL` to simulate failures for all Aurora Replicas in the DB cluster. Specify `TO` and the name of the Aurora Replica to simulate a failure of a single Aurora Replica. The default failure type is `TO ALL`.
- **`quantity`—**The amount of time for which to simulate the Aurora Replica failure. The interval is an amount followed by a time unit. The simulation will occur for that amount of the specified unit. For example, `20 MINUTE` will result in the simulation running for 20 minutes.

  Note
  Take care when specifying the time interval for your Aurora Replica failure event. If you specify too long of a time interval, and your master instance writes a large amount of data during the failure event, then your Aurora DB cluster might assume that your Aurora Replica has crashed and replace it.

Testing a Disk Failure

You can simulate a disk failure for an Aurora DB cluster using the `ALTER SYSTEM SIMULATE DISK FAILURE` fault injection query.

During a disk failure simulation, the Aurora DB cluster randomly marks disk segments as faulting. Requests to those segments will be blocked for the duration of the simulation.

Syntax

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK FAILURE

[ IN DISK index | NODE index ]

FOR INTERVAL quantity [ YEAR | QUARTER | MONTH | WEEK| DAY | HOUR | MINUTE |
SECOND ];
```

Options

This fault injection query takes the following parameters:

- **`percentage_of_failure`—**The percentage of the disk to mark as faulting during the failure event. This value can be a double between 0 and 100. If you specify 0, then none of the disk is marked as faulting. If you specify 100, then the entire disk is marked as faulting.
- **`DISK index` or `NODE index`—**A specific disk or node to simulate the failure event for. If you exceed the range of indexes for the disk or node, you will receive an error that tells you the maximum index value that you can specify.
- **`quantity`—**The amount of time for which to simulate the disk failure. The interval is an amount followed by a time unit. The simulation will occur for that amount of the specified unit. For example, `20 MINUTE` will result in the simulation running for 20 minutes.

Testing Disk Congestion

You can simulate a disk failure for an Aurora DB cluster using the `ALTER SYSTEM SIMULATE DISK CONGESTION` fault injection query.

During a disk congestion simulation, the Aurora DB cluster randomly marks disk segments as congested. Requests to those segments will be delayed between the specified minimum and maximum delay time for the duration of the simulation.

Syntax

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT

DISK CONGESTION BETWEEN minimum AND maximum MILLISECONDS

[ IN DISK index | NODE index ]

FOR INTERVAL quantity [ YEAR | QUARTER | MONTH | WEEK| DAY | HOUR | MINUTE |
SECOND ];
```

Options

This fault injection query takes the following parameters:

- **`percentage_of_failure`—**The percentage of the disk to mark as congested during the failure event. This value can be a double between 0 and 100. If you specify 0, then none of the disk is marked as congested. If you specify 100, then the entire disk is marked as congested.
- **`DISK index` or `NODE index`—**A specific disk or node to simulate the failure event for. If you exceed the range of indexes for the disk or node, you will receive an error that tells you the maximum index value that you can specify.
- **`minimum` and `maximum`—**The minimum and maximum amount of congestion delay, in milliseconds. Disk segments marked as congested will be delayed for a random amount of time within the range of the minimum and maximum amount of milliseconds for the duration of the simulation.
- **`quantity`—**The amount of time for which to simulate the disk congestion. The interval is an amount followed by a time unit. The simulation will occur for that amount of the specified time unit. For example, `20 MINUTE` will result in the simulation running for 20 minutes.

Best Practices with Amazon Aurora

This topic includes information on best practices and options for using or migrating data to an Amazon Aurora DB cluster.

Topics

- Determining Which DB Instance You Are Connected To (p. 434)
- Using Amazon Aurora to Scale Reads for Your MySQL Database (p. 434)
- Using Amazon Aurora for Disaster Recovery with Your MySQL Databases (p. 436)
- Migrating from MySQL to Amazon Aurora with Reduced Downtime (p. 437)

Determining Which DB Instance You Are Connected To

You can determine which DB instance in an Aurora DB cluster that a connection is connected to by checking the `innodb_read_only` global variable, as shown in the following example.

```
SHOW GLOBAL VARIABLES LIKE 'innodb_read_only';
```

The `innodb_read_only` variable will be set to `ON` if you are connected to an Aurora Replica and `OFF` if you are connected to the primary instance.

This can be helpful if you want to add logic to your application code to balance the workload or to ensure that a write operation is using the correct connection.

Using Amazon Aurora to Scale Reads for Your MySQL Database

You can use Amazon Aurora with your MySQL DB instance to take advantage of the read scaling capabilities of Amazon Aurora and expand the read workload for your MySQL DB instance. To use Aurora to read scale your MySQL DB instance, create an Amazon Aurora DB cluster and make it a replication slave of your MySQL DB instance. This applies to an Amazon RDS MySQL DB instance, or a MySQL database running external to Amazon RDS.

For information on creating an Amazon Aurora DB cluster, see Creating an Amazon Aurora DB Cluster (p. 398).

When you set up replication between your MySQL DB instance and your Amazon Aurora DB cluster, be sure to follow these guidelines:

- Use the Amazon Aurora DB cluster endpoint address when you reference your Amazon Aurora DB cluster. If a failover occurs, then the Aurora Replica that is promoted to the primary instance for the Aurora DB cluster will continue to use the DB cluster endpoint address.
- Maintain the binlogs on your master instance until you have verified that they have been applied to the Aurora replica. This maintenance ensures that you can restore your master instance in the event of a failure.

Important
There is a known issue when Amazon Aurora DB cluster is the replication slave where replication will pause unexpectedly. In this case, the CloudWatch **Replica Lag** will continue to grow. If this occurs, you must restore the Amazon Aurora DB cluster from the most recent snapshot and set up replication with the restored DB cluster as the new replication slave.

Note
The permissions required to start replication on an Amazon Aurora DB cluster are restricted and not available to your Amazon RDS master user. Because of this, you must use the Amazon RDS mysql.rds_set_external_master (p. 178) and mysql.rds_start_replication (p. 180) commands to set up replication between your Amazon Aurora DB cluster and your MySQL DB instance.

Start Replication Between an External Master Instance and a MySQL DB Instance on Amazon RDS

1. Make the source MySQL DB instance read-only:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Run the `SHOW MASTER STATUS` command on the source MySQL DB instance to determine the binlog location. You will receive output similar to the following example:

```
File                          Position
-----------------------------------
 mysql-bin-changelog.000031      107
-----------------------------------
```

3. Copy the database from the external MySQL DB instance to the Amazon Aurora DB cluster using `mysqldump`. For very large databases, you might want to use the procedure in Importing Data to an Amazon RDS MySQL DB Instance with Reduced Downtime (p. 147).

```
mysqldump   databases <database_name>   single-transaction   compress   order-
by-primary
     –u <local_user> -p<local_password> | mysql   host=aurora_cluster_end
point_address –-port=3306 –u <RDS_user_name> –p<RDS_password>
```

   Note
   Make sure there is not a space between the `-p` option and the entered password.

   Use the `host`, `user (-u)`, `port` and `–p` options in the `mysql` command to specify the hostname, username, port, and password to connect to your Aurora DB cluster. The host name is the DNS name from the Amazon Aurora DB cluster endpoint, for example, `mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the cluster details in the Amazon RDS Management Console.

4. Make the source MySQL DB instance writeable again:

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

   For more information on making backups for use with replication, go to Backing Up a Master or Slave by Making It Read Only in the MySQL documentation.

5. In the Amazon RDS Management Console, add the IP address of the server that hosts the source MySQL database to the VPC security group for the Amazon Aurora DB cluster. For more information on modifying a VPC security group, go to Security Groups for Your VPC in the *Amazon Virtual Private Cloud User Guide*.

   You might also need to configure your local network to permit connections from the IP address of your Amazon Aurora DB cluster, so that it can communicate with your source MySQL instance. To find the IP address of the Amazon Aurora DB cluster, use the `host` command:

```
host <aurora_endpoint_address>
```

The host name is the DNS name from the Amazon Aurora DB cluster endpoint.

6. Using the client of your choice, connect to the external MySQL instance and create a MySQL user that will be used for replication. This account is used solely for replication and must be restricted to your domain to improve security. The following is an example:

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>';
```

7. For the external MySQL instance, grant `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges to your replication user. For example, to grant the `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges on all databases for the '`repl_user`' user for your domain, issue the following command:

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydo
main.com' IDENTIFIED BY '<password>';
```

8. Take a manual snapshot of the Aurora DB cluster that will be the replication slave prior to setting up replication. If you need to reestablish replication with the DB cluster as a replication slave, you can restore the Aurora DB cluster from this snapshot instead of having to import the data from your MySQL DB instance into a new Aurora DB cluster.
9. Make the Amazon Aurora DB cluster the replica. Connect to the Amazon Aurora DB cluster as the master user and identify the source MySQL database as the replication master by using the mysql.rds_set_external_master (p. 178) command. Use the master log file name and master log position that you determined in Step 2. The following is an example:

```
CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306,
    'repl_user', '<password>', 'mysql-bin-changelog.000031', 107, 0);
```

10. On the Amazon Aurora DB cluster, issue the mysql.rds_start_replication (p. 180) command to start replication:

```
CALL mysql.rds_start_replication;
```

After you have established replication between your source MySQL DB instance and your Amazon Aurora DB cluster, you can add Aurora Replicas to your Amazon Aurora DB cluster. You can then connect to the Aurora Replicas to read scale your data. For information on creating an Aurora Replica, see Creating an Aurora Replica Using the Console (p. 405).

Using Amazon Aurora for Disaster Recovery with Your MySQL Databases

You can use Amazon Aurora with your MySQL DB instance to create an off-site backup for disaster recovery. To use Aurora for disaster recovery of your MySQL DB instance, create an Amazon Aurora DB cluster and make it a replication slave of your MySQL DB instance. This applies to an Amazon RDS MySQL DB instance, or a MySQL database running external to Amazon RDS.

Important
When you set up replication between a MySQL DB instance and an Amazon Aurora DB cluster, the replication is not managed by Amazon RDS. You must monitor the replication to ensure that it remains healthy and repair it if necessary.

For instructions on how to create an Amazon Aurora DB cluster and make it a replication slave of your MySQL DB instance, follow the procedure in Using Amazon Aurora to Scale Reads for Your MySQL Database (p. 434).

Migrating from MySQL to Amazon Aurora with Reduced Downtime

When importing data from a MySQL database that supports a live application to an Amazon Aurora DB cluster, you can use the procedure documented in Importing Data to an Amazon RDS MySQL DB Instance with Reduced Downtime (p. 147) to reduce the amount of time that service to your data is interrupted in order to migrate your data to Aurora. The procedure can especially help if you are working with a very large database, because you can reduce the cost of the import by minimizing the amount of data that is passed across the network to AWS.

The procedure lists steps to transfer a copy of your database data to an Amazon EC2 instance and import the data into a new Amazon RDS MySQL DB instance. Because Amazon Aurora is compatible with MySQL, you can instead use an Amazon Aurora DB cluster for the target Amazon RDS MySQL DB instance.

Appendix: DB Cluster and DB Instance Parameters

You manage your Amazon Aurora DB cluster in the same way that you manage other Amazon RDS DB instances, by using parameters in a DB parameter group. Amazon Aurora differs from other DB engines in that you have a cluster of DB instances. As a result, some of the parameters that you use to manage your Amazon Aurora DB cluster apply to the entire cluster, while other parameters apply only to a particular DB instance in the DB cluster.

Cluster-level parameters are managed in DB cluster parameter groups. Instance-level parameters are managed in DB parameter groups. Although each instance in an Aurora DB cluster is compatible with the MySQL database engine, some of the MySQL database engine parameters must be applied at the cluster level, and are managed using DB cluster parameter groups. Cluster-level parameters are not found in the DB parameter group for an instance in an Aurora DB cluster and are listed later in this topic.

You can manage both cluster-level and instance-level parameters using the Amazon RDS console, the AWS CLI, or the Amazon RDS API. There are separate commands for managing cluster-level parameters and instance-level parameters. For example, you can use the ModifyDBClusterParameterGroup API action to manage cluster-level parameters in a DB cluster parameter group and use the ModifyDBParameterGroup to manage instance-level parameters in a DB parameter group for a DB instance in a DB cluster.

For more information on DB parameter groups, see Working with DB Parameter Groups (p. 523).

> **Note**
> Some MySQL database engine parameters that are available to RDS MySQL DB instances are not available to instances in an Aurora DB cluster. For details on the differences between an Aurora instance and a MySQL DB instance, see Comparison of Amazon RDS for Aurora and Amazon RDS for MySQL (p. 397).

Cluster-level parameters

The following table shows all of the parameters that apply to the entire Amazon Aurora DB cluster. Parameters that are not included in this list apply to individual DB instances in a DB cluster.

- auto_increment_increment
- auto_increment_offset
- binlog_checksum
- binlog_format
- binlog_row_image
- binlog_rows_query_log_events
- character_set_database
- character_set_filesystem
- collation_database
- completion_type
- default_storage_engine
- default_tmp_storage_engine
- default-time-zone
- foreign_key_checks
- gtid_mode
- ignore_default_storage_engine_errors
- init_slave
- innodb_autoinc_lock_mode
- innodb_checksum_algorithm
- innodb_checksums
- innodb_cmp_per_index_enabled
- innodb_commit_concurrency
- innodb_data_file_path
- innodb_data_home_dir
- innodb_doublewrite
- innodb_fast_shutdown
- innodb_file_per_table
- innodb_flush_log_at_trx_commit
- innodb_ft_max_token_size
- innodb_ft_min_token_size
- innodb_ft_num_word_optimize
- innodb_ft_sort_pll_degree
- innodb_online_alter_log_max_size
- innodb_optimize_fulltext_only
- innodb_page_size
- innodb_print_all_deadlocks
- innodb_purge_batch_size
- innodb_purge_threads
- innodb_rollback_on_timeout
- innodb_rollback_segments
- innodb_spin_wait_delay
- innodb_strict_mode
- innodb_support_xa
- innodb_sync_array_size
- innodb_sync_spin_loops
- innodb_table_locks

- innodb_undo_directory
- innodb_undo_logs
- innodb_undo_tablespaces
- lc_messages
- lc_messages_dir
- lc_time_names
- lower_case_file_system
- lower_case_table_names
- master_verify_checksum
- master-info-repository
- report_host
- report_password
- report_port
- report_user
- server_id
- skip_name_resolve
- skip-character-set-client-handshake
- skip-slave-start
- ssl_ca
- ssl_capath
- ssl_cert
- ssl_cipher
- ssl_crl
- ssl_crlpath
- ssl_key
- storage_engine
- sync_frm
- system_time_zone
- time_zone
- unique_checks

Amazon RDS DB Instance Lifecycle

The lifecycle of an Amazon RDS DB instance includes creating, modifying, maintaining and upgrading, performing backups and restores, rebooting, and deleting the instance. This section provides information on and links to more about these processes.

Many tasks you need to perform on a DB instance, such as rebooting or renaming, are performed the same way for all DB engines. Tasks such as creating a DB instance for a specific DB engine, connecting to a DB instance, and importing data into that DB instance are all tasks that are specific to each DB engine.

The following table shows the Amazon RDS operations you are most likely to use, and provides links to procedural instruction and examples. Some of these topics are in this section, and others appear in other sections of the Amazon RDS documentation.

Topic	Section in Amazon RDS User Guide
Creating a DB instance (DB engine specific)	Creating a DB Instance Running the MySQL Database Engine (p. 129) Creating a DB Instance Running the Oracle Database Engine (p. 205) Creating a DB Instance Running the PostgreSQL Database Engine (p. 363) Creating a DB Instance Running the SQL Server Database Engine (p. 313)
Backup/Restore	
Backing up a DB instance	Backing Up and Restoring (p. 495)
Creating a snapshot of a DB instance	Backing Up and Restoring (p. 495)
Performing a point-in-time restore	Restoring a DB Instance to a Specified Time (p. 508)
Modifying	

Topic	Section in Amazon RDS User Guide
Modifying a DB instance (DB engine specific)	Modifying a DB Instance Running the MySQL Database Engine (p. 140) Modifying a DB Instance Running the Oracle Database Engine (p. 216) Modifying a DB Instance Running the PostgreSQL Database Engine (p. 374) Modifying a DB Instance Running the SQL Server Database Engine (p. 328)
Renaming a DB instance	Renaming a DB Instance (p. 457)
Changing the storage type	Working with Storage Types (p. 464)
Tagging a DB instance	Tagging Amazon RDS Resources (p. 486)
Maintaining and Upgrading	
Performing maintenance on a DB instance	Amazon RDS Maintenance Window (p. 451)
Upgrading a DB instance	Upgrading and Amazon RDS Resource Maintenance (p. 442)
Rebooting a DB instance	Rebooting a DB Instance (p. 462)
Deleting a DB instance	Deleting a DB Instance (p. 459)

Upgrading and Amazon RDS Resource Maintenance

Periodically, the Amazon RDS system performs maintenance on Amazon RDS resources, such as DB instances. Maintenance can range from a DB instance's operating system (OS) updates and DB engine upgrades to changing size or storage for an instance. A DB engine upgrade can consist of either a major or minor database version upgrade.

You can choose to apply maintenance items on a DB instance at your convenience instead of waiting for the maintenance process initiated by AWS during your maintenance window. You can view whether a maintenance update is available for your DB instance both on the RDS console and by using the Amazon RDS API. If an update is available, you can choose to do one of the following:

- Have the maintenance items applied immediately.
- Schedule the maintenance items to be applied during your next maintenance window.

  Note
  The maintenance window setting determines when pending operations are to be started and does not limit the total execution time of these operations. In other words, pending operations will be initiated during the configured maintenance window but they are not guaranteed to finish before the maintenance window closes.
- Defer the maintenance items.

Certain maintenance items will be marked as **Required** in the **Maintenance** column in the Amazon RDS console. These updates cannot be deferred indefinitely. If you choose to defer a required update, you will receive a communication from AWS that notifies you of the time at which the update will be performed on your DB instance. Other updates will be marked as **Available**. You can defer these maintenance items indefinitely and the update will not be applied to your DB instance.

Maintenance items require that Amazon RDS take your DB instance offline for a short time. Maintenance that requires your DB instance to be offline include scale compute operations, which generally take only a few minutes from start to finish, and required operating system or database patching. Required patching is automatically scheduled only for patches that are related to security and instance reliability. Such patching occurs infrequently (typically once every few months) and seldom requires more than a fraction of your maintenance window.

If you don't specify a preferred weekly maintenance window when creating your DB instance, a 30-minute default value is assigned. If you want to change when maintenance is performed on your behalf, you can do so by modifying your DB instance in the AWS Management Console or by using the `ModifyDBInstance` API. Each of your DB instances can have different preferred maintenance windows, if you so choose.

Running your DB instance as a Multi-AZ deployment can further reduce the impact of a maintenance event, because Amazon RDS will conduct maintenance by following these steps:

1. Perform maintenance on the standby.
2. Promote the standby to primary.
3. Perform maintenance on the old primary, which becomes the new standby.

Note
When you modify the database engine for your DB instance in a Multi-AZ deployment, then Amazon RDS upgrades both the primary and secondary DB instances at the same time. In this case, the database engine for the entire Multi-AZ deployment is shut down during the upgrade.

For more information on Multi-AZ deployments, see High Availability (Multi-AZ) (p. 71).

Topics

- Operating System Updates for a DB Instance (p. 443)
- Upgrading Database Versions for a DB Instance (p. 446)
- Amazon RDS Maintenance Window (p. 451)

Operating System Updates for a DB Instance

Operating system (OS) updates for Amazon RDS DB instances are most often updates for security or reliability reasons. You can decide when Amazon RDS will apply OS updates by using the RDS console, command line interface (CLI), or API.

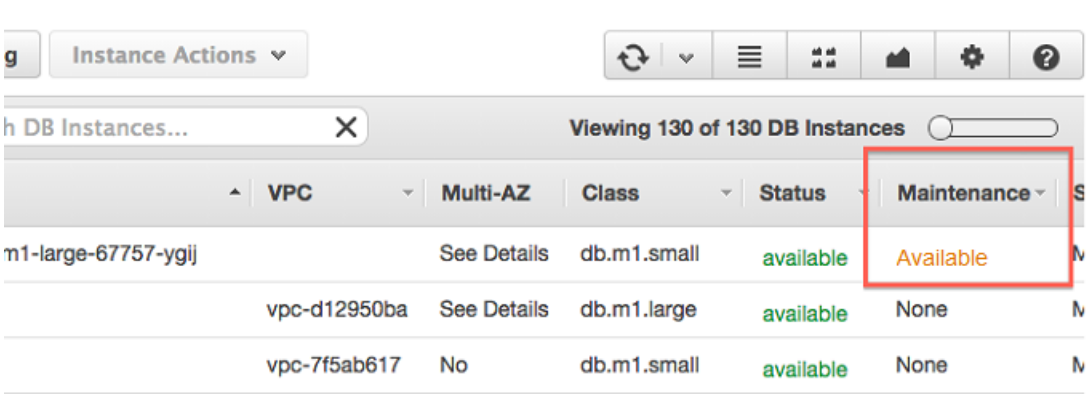
You can choose to apply OS updates on a DB instance at your convenience instead of waiting for the maintenance process initiated by AWS during your maintenance window. You can view whether an OS update is available for your DB instance both on the Amazon RDS console and by using the Amazon RDS CLI or API. If an update is available, it will be indicated by the word **Available** in the **Maintenance** column for the DB instance on the Amazon RDS console. For OS updates that are marked **Available**, you can choose to do one of the following:

- Have the OS update applied immediately.
- Schedule the OS update to be applied during your next maintenance window.

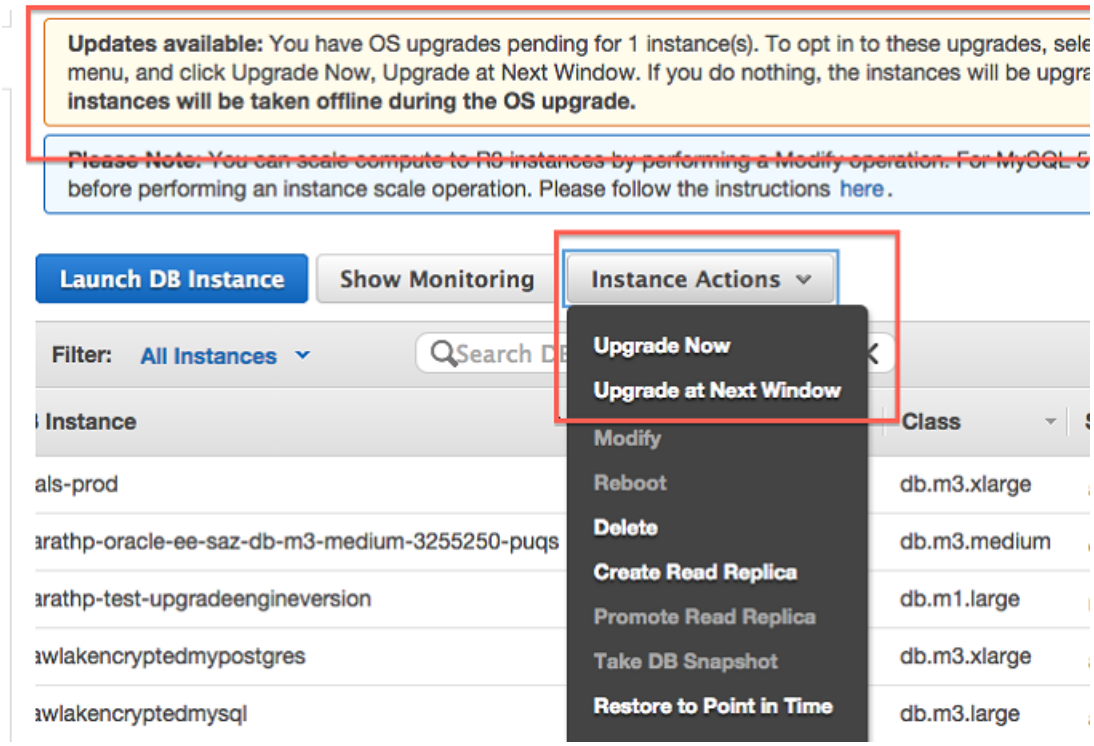
  Note
  The maintenance window setting determines when pending operations are to be started and does not limit the total execution time of these operations. In other words, pending operations will be initiated during the configured maintenance window but they are not guaranteed to finish before the maintenance window closes.
- Defer the OS update.

Certain OS updates will be marked as **Required** in the **Maintenance** column in the Amazon RDS console. These updates cannot be deferred indefinitely. If you choose to defer a required update, you will receive a communication from AWS that notifies you of the time at which the update will be performed on your DB instance. Other updates will be marked as **Available**. You can defer these OS updates indefinitely and the update will not be applied to your DB instance.

If you use the Amazon RDS console, it will indicate when an operating system update is either available or required for your DB instance. For example, the following screenshot shows that an OS update is available:



The **Maintenance** column indicates whatever option you select. For example, the following screenshot shows that the selected DB instance can be updated either immediately or during the DB instance's next maintenance window:



AWS Management Console

To manage an OS update for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Instances**.
3. Click the check box for the DB instance that has a required operating system update.
4. Click **Instance Actions** and click one of the following:

   - **Upgrade Now**
   - **Upgrade at Next Window**

     Note
     If you choose the **Upgrade at Next Window** option, and later want to delay the OS update, you can click **Instance Actions** and then select **Defer Upgrade**.

CLI

To apply a pending OS update to a DB instance

- Use the rds-apply-pending-maintenance-action command to apply pending maintenance actions.

Example

```
PROMPT> rds-apply-pending-maintenance-action arn:aws:rds:us-west-
2:001234567890:db:mysql-db -a os-upgrade -o immediate
```

To return a list of resources that have at least one pending OS update

- Use the rds-describe-pending-maintenance-actions command to list all pending maintenance actions.

Example

```
PROMPT> rds-describe-pending-maintenance-actions arn:aws:rds:us-west-
2:001234567890:db:mysql-db
```

API

To apply an OS update to a DB instance

- Call ApplyPendingMaintenanceAction.

Example

```
https://rds.us-west-2.amazonaws.com/
   ?Action=ApplyPendingMaintenanceAction
   &ResourceIdentifier=arn:aws:rds:us-east-1:123456781234:db:my-instance
   &ApplyAction=os-upgrade
   &OptInType=immediate
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-10-31
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20141216/us-west-2/rds/aws4_request
   &X-Amz-Date=20140421T194732Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=6e25c542bf96fe24b28c12976ec92d2f856ab1d2a158e21c35441a736e4fde2b
```

To return a list of resources that have at least one pending OS update

- Call DescribePendingMaintenanceActions.

Example

```
https://rds.us-west-2.amazonaws.com/
   ?Action=DescribePendingMaintenanceActions
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-10-31
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20141216/us-west-2/rds/aws4_request
   &X-Amz-Date=20140421T194732Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=6e25c542bf96fe24b28c12976ec92d2f856ab1d2a158e21c35441a736e4fde2b
```

Upgrading Database Versions for a DB Instance

Database version upgrades consist of major and minor version upgrades. You control when to upgrade your DB instance to a new version supported by Amazon RDS. This level of control helps you maintain compatibility with specific database versions and test new versions with your application before deploying in production. When you are ready, you can perform version upgrades at the times that best fit your schedule. You can also schedule them to be applied automatically.

Amazon RDS supports the following changes to your operating system or database engine version:

- **OS Updates**, which can be manually or automatically applied. In some cases, an OS update must be applied before you can perform a DB engine version upgrade. For more information, see Operating System Updates for a DB Instance (p. 443).
- **Minor DB engine version upgrades**, which can be manually or automatically applied. You can specify whether minor DB engine version upgrades are applied to your DB instance using the **AutoMinorVersionUpgrade** option. For more information, see Minor DB Engine Version Upgrades for a DB Instance (p. 447)
- **Major DB engine version upgrades**, which can only be manually applied. Major DB version upgrades present a risk to backward-compatibility and must be fully tested before being applied. Amazon RDS does not automatically apply major DB version upgrades. For more information, see Major DB Engine Version Upgrades for a DB Instance (p. 447)

Amazon RDS currently supports the following DB engine in-place major version upgrades:

- MySQL 5.1 to MySQL 5.5
- MySQL 5.5 to MySQL 5.6
- Microsoft SQL Server 2008 to SQL Server 2012

Important
In some cases, you must first perform an OS update before you can perform a major database version upgrade. For example, MySQL DB instances created before April 24, 2014, are required to update their OS before upgrading to MySQL version 5.6. In this case, the DB instance will show that it has an **Available** OS update. For more information on OS updates, see Operating System Updates for a DB Instance (p. 443) earlier in this topic.

Major DB Engine Version Upgrades for a DB Instance

Major database version upgrades can contain changes that are not backward-compatible with existing applications. As a result, Amazon RDS does not apply major version upgrades automatically; you must manually modify your DB instance. You should thoroughly test any upgrade before applying it to your production instances.

You use the Amazon RDS modify operation to perform a major version upgrade of a DB instance. You can also use the modify operation to manually apply a minor version upgrade. To perform a major version upgrade, follow the instructions in Upgrading the Database Version of a DB Instance (p. 449) later in this topic.

Major DB Engine Version Upgrades for a MySQL DB Instance

To perform a major version upgrade for a MySQL version 5.1 DB instance on Amazon RDS to MySQL version 5.6, you must first perform any **Available** OS updates, then upgrade your DB instance to MySQL version 5.5, and then upgrade your DB instance to MySQL version 5.6. MySQL DB instances created before April 24, 2014 will show an **Available** OS update (unless the update has been applied).

Follow the instructions in Operating System Updates for a DB Instance (p. 443) to update the OS for your DB instance. Once you have updated the OS for your DB instance, follow the instructions in Upgrading the Database Version of a DB Instance (p. 449) later in this topic to upgrade the engine version for your DB instance.

Major DB Engine Version Upgrades for a PostgreSQL DB Instance

Amazon RDS does not currently support an in-place upgrade from a PostgreSQL 9.3.x DB instance to a PostgreSQL 9.4.1 DB instance. However, you can migrate a database from a PostgreSQL 9.3.5 DB instance to a PostgreSQL 9.4.1 DB instance by using the following steps:

1. Create a new PostgreSQL 9.4.1 DB instance
2. Create a backup of your existing PostgreSQL 9.3.5 database using *pg_dump*
3. Import the dump file into your PostgreSQL 9.4.1 DB instance using *pg_restore*

Be sure to test your application against the new version of PostgreSQL before going into production.

Minor DB Engine Version Upgrades for a DB Instance

Minor database version upgrades generally involve less compatibility risk than major version upgrades. By default, your DB instance will automatically be upgraded to new minor versions as they are supported by Amazon RDS. This patching occurs during your scheduled maintenance window, and is announced on the Amazon RDS Community Forum in advance. To turn off automatic version upgrades in the Amazon RDS console, modify your DB instance and set **AutoMinorVersionUpgrade** to `No`.

If you set **AutoMinorVersionUpgrade** to `No`, you can manually upgrade a DB instance to a supported minor version release by modifying the DB instance and specifying the new minor version number.

Testing an Upgrade

Before you perform a major version upgrade on your DB instance, you should thoroughly test both your database and the host application for compatibility. We suggest you do the following:

1. Review the upgrade documentation for the new version of the database engine to see if there are compatibility issues that might affect your database or applications:
   - MySQL 5.5 Upgrade Documentation
   - MySQL 5.6 Upgrade Documentation

- Upgrade to SQL Server 2012

2. If your DB instance is a member of a custom DB parameter group, you need to create a new DB parameter group with your existing settings that is compatible with the new major version. Specify the new DB parameter group when you upgrade your test instance, so that your upgrade testing ensures that it works correctly. For more information about creating a DB parameter group, see Working with DB Parameter Groups (p. 523).
3. Create a DB snapshot of the DB instance to be upgraded. For more information, see Creating a DB Snapshot (p. 499).
4. Restore the DB snapshot to create a new test DB instance. For more information, see Restoring From a DB Snapshot (p. 501).
5. Modify this new test DB instance to upgrade it to the new version, using one of the methods detailed following. If you created a new parameter group in step 2, specify that parameter group.
6. Evaluate the storage used by the upgraded instance to determine if the upgrade requires additional storage.
7. Run as many of your quality assurance tests against the upgraded DB instance as needed to ensure that your database and application work correctly with the new version. Implement any new tests needed to evaluate the impact of any compatibility issues you identified in step 1. Test all stored procedures and functions. Direct test versions of your host applications to the upgraded DB instance.
8. If all tests pass, then perform the upgrade on your production DB instance. We suggest you do not allow write operations to the DB instance until you can confirm that everything is working correctly.

Considerations for DB Instance Upgrades

If your DB instance is using read replication, you must upgrade all of the Read Replicas before upgrading the source instance. If the DB instance is in a Multi-AZ deployment, both the primary and standby replicas are upgraded, and the instance might not be available until the upgrade is complete.

Amazon RDS takes two DB snapshots during the upgrade process. The first DB snapshot is of the DB instance before any upgrade changes have been made. If the upgrade doesn't work for your databases, you can restore this snapshot to create a DB instance running the old version. The second DB snapshot is taken when the upgrade completes.

If Amazon RDS encounters any issues when upgrading SQL Server and needs to roll back, Amazon RDS restores your DB instance from the first DB snapshot.

For MySQL upgrades, the first snapshot is a user snapshot that is retained until you delete it. For SQL Server upgrades, the first snapshot is a system snapshot that is retained only until the end of the backup retention period.

MySQL major version upgrades typically complete in about 10 minutes, and SQL Server upgrades typically complete in about 30 minutes. Upgrades might take longer, depending on the class of DB instance, or whether the instance follows the operational guidelines in Best Practices for Amazon RDS (p. 54). If you are upgrading a DB instance from the Amazon RDS console, the status of the DB instance will indicate when the upgrade is complete. If you are using the CLI, use the **rds-describe-db-instance** command and check the *Status* value.

After the upgrade is complete, you cannot revert to the previous version of the database engine. If you want to return to the previous version, restore the first DB snapshot taken to create a new DB instance.

In addition to applying major version upgrades to instances of MySQL and SQL Server, you can manually apply minor version upgrades to instances of MySQL or Oracle Database. You can also specify that minor version upgrades be applied automatically to instances of MySQL and Oracle. For more information about managing minor version upgrades, see MySQL on Amazon RDS Versions (p. 120) or Oracle Version Management (p. 203).

Note
During a major version upgrade of SQL Server, the **Free Storage Space** and **Disk Queue Depth** metrics will display `-1`. After the upgrade is complete, both metrics will return to normal.
During a major version upgrade of MySQL, Amazon RDS runs the MySQL binary `mysql_upgrade` to upgrade tables, if required. Also, Amazon RDS will empty the slow_log and general_log tables during a major version upgrade. If you need to preserve log information, then you will need to save the log contents before the major version upgrade.

Upgrading the Database Version of a DB Instance

The following procedures cover the process of upgrading the database version of a DB instance.

AWS Management Console

To apply a DB engine major version upgrade to a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Instances**.
3. Click the check box for the DB instance that you want to upgrade.
4. Click **Instance Actions** and click **Modify**.
5. In the **DB Engine Version** box, click the new version.
6. To upgrade immediately, click to select the **Apply Immediately** check box. To delay the upgrade to the next maintenance window, make sure this check box is clear.
7. Click **Continue**.
8. Review the modification summary information. To proceed with the upgrade, click **Modify DB Instance**. To cancel the upgrade, click the **X** in the upper right corner.

CLI

To apply a DB engine major version upgrade to a DB instance

- Use the CLI command rds-modify-db-instance specifying the DB instance identifier and using the following parameters:
  - `--engine-version` = the new DB engine version
  - `--allow-major-version-upgrade` = `true`
  - `--apply-immediately` = `true` to upgrade immediately, or `false` to delay the upgrade until the next maintenance window

Example

```
PROMPT>rds-modify-db-instance SQLServer1 --engine-version 11.00.2100.60.v1
 --allow-major-version-upgrade true --apply-immediately true
```

API

To apply a DB engine major version upgrade to a DB instance

- Call ModifyDBInstance with the following parameters:

- *DBInstanceIdentifier* = the identifier of the instance to be upgraded
- *EngineVersion* = the new DB engine version
- *AllowMajorVersionUpgrade* = `true`
- *ApplyImmediately* = `true` to upgrade immediately, or `false` to delay the upgrade until the next maintenance window

Example

```
https://rds.amazonaws.com/
 ?Action=ModifyDBInstance
 &DBInstanceIdentifier=MySQL-Instance1
 &EngineVersion=5.5.31
 &AllowMajorVersionUpgrade=true
 &ApplyImmediately=true
```

Upgrading a MySQL Database with Reduced Downtime

If your MySQL DB instance is currently in use with a production application, you can use the following procedure to upgrade the database version for your DB instance and reduce the amount of downtime for your application.

To upgrade an MySQL database while a DB instance is in use

1. Using the Amazon RDS console, create a Read Replica of your MySQL 5.5 DB instance. This process will create an upgradable copy of your database.
   a. On the console, click **Instances** and click the DB instance that you want to upgrade.
   b. Click **Instance Actions** and click **Create Read Replica**.
   c. Provide a value for **DB Instance Identifier** for your Read Replica and ensure that the DB instance **Class** and other settings match your MySQL 5.5 DB instance.
   d. Click **Yes, Create Read Replica**.
2. When the Read Replica has been created and **Status** shows **available**, upgrade the Read Replica to MySQL 5.6.
   a. On the console, click **Instances** and click the Read Replica that you just created.
   b. Click **Instance Actions** and click **Modify**.
   c. In the **DB Engine Version** box, select the MySQL 5.6 version to upgrade to and click the **Apply Immediately** check box. Click **Continue**.
   d. Click **Modify DB Instance** to start the upgrade.
3. When the upgrade is complete and **Status** shows **available**, verify that the upgraded Read Replica is up to date with the master MySQL 5.5 DB instance. You can do this by connecting to the Read Replica and issuing the `SHOW SLAVE STATUS` command. If the `Seconds_Behind_Master` field is `0`, then replication is up to date.
4. Make your MySQL 5.6 Read Replica a master DB instance.

Important
When you promote your MySQL 5.6 Read Replica to a stand-alone, single-AZ DB instance, it will no longer be a replication slave to your MySQL 5.5 DB instance. We recommend that you promote your MySQL 5.6 Read Replica during a maintenance window when your source MySQL 5.5 DB instance is in read-only mode and all write operations are suspended. When the promotion is completed, you can direct your write operations to the upgraded MySQL 5.6 DB instance to ensure that no write operations are lost.
In addition, we recommend that prior to promoting your MySQL 5.6 Read Replica you perform all necessary data definition language (DDL) operations, such as creating indexes, on the MySQL 5.6 Read Replica. This approach will avoid any negative effects on the performance of the MySQL 5.6 Read Replica after it has been promoted.

a. On the console, click **Instances** and click the Read Replica that you just upgraded.
b. Click **Instance Actions** and click **Promote Read Replica**.
c. Enable automated backups for the Read Replica instance. For more information, see Working With Automated Backups (p. 496).

   Click **Continue**.
d. Click **Yes, Promote Read Replica**.

5. You now have an upgraded version of your MySQL database. At this point, you can direct your applications to the new MySQL 5.6 DB instance, add Read Replicas, set up Multi-AZ support, and so on.

Related Topics

- Amazon RDS Maintenance Window (p. 451)
- Amazon RDS DB Instances (p. 64)
- DB Instance Class (p. 65)

Amazon RDS Maintenance Window

Most maintenance occurs during a user-definable maintenance window. You can think of the maintenance window as an opportunity to control when DB instance modifications (such as implementing pending changes to storage or CPU class for the DB instance) and software patching occur, in the event either are requested or required. If a maintenance event is scheduled for a given week, it will be initiated and completed at some point during the 30 minute maintenance window you identify.

Amazon RDS allows you to choose when to upgrade your DB software and when you upgrade the underlying operating system. Upgrades to the operating system are most often for security issues and should be done as soon as possible. This gives you the ability to see ahead of time when a given required maintenance update will be applied to their instances, as well as the ability to opt in to the maintenance ahead of the scheduled start time.

The 30-minute maintenance window is selected at random from an 8-hour block of time per region. If you don't specify a preferred maintenance window when you create the DB instance, Amazon RDS assigns a 30-minute maintenance window on a randomly selected day of the week.

The following table lists the time blocks for each region from which the default maintenance windows are assigned.

Region	Time Block
US East (N. Virginia) region	03:00-11:00 UTC
US West (N. California) region	06:00-14:00 UTC
US West (Oregon) region	06:00-14:00 UTC
EU (Ireland) region	22:00-06:00 UTC
EU (Frankfurt) Region	23:00-07:00 UTC
Asia Pacific (Tokyo) Region	13:00-21:00 UTC
Asia Pacific (Sydney) Region	12:00-20:00 UTC
Asia Pacific (Singapore) Region	14:00-22:00 UTC
South America (São Paulo) Region	00:00-08:00 UTC
AWS GovCloud (US) Region	06:00-14:00 UTC

Adjusting the Preferred Maintenance Window

Every DB instance has a weekly maintenance window during which any system changes are applied. If you don't specify a preferred maintenance window when you create the DB Instance, Amazon RDS assigns a 30-minute maintenance window on a randomly selected day of the week. The 30-minute maintenance window is selected at random from an 8-hour block of time per region. For more information about what changes are applied during the maintenance window, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 455)

Amazon RDS gives you the ability to see ahead of time when a given required maintenance update will be applied to your instances, as well as the ability to opt in to the maintenance ahead of the scheduled start time.

The maintenance window should fall at the time of lowest usage and thus might need modification from time to time. Your DB instance will only be unavailable during this time if the system changes, such as a scale storage operation or a change in DB instance class, are being applied and require an outage, and only for the minimum amount of time required to make the necessary changes.

In the following example, you adjust the preferred maintenance window for a DB Instance.

For the purpose of this example, we assume that the DB instance named *mydbinstance* exists and has a preferred maintenance window of "Sun:05:00-Sun:06:00" UTC.

AWS Management Console

To adjust the preferred maintenance window

1. Launch the AWS Management Console.
   a. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
   b. Click on the **DB Instances** link in the Navigation panel on the left side of the console display.

      The **My Instances** list appears.
   c. Right-click on the **DB Instance** in the **My DB Instances** list and select **Modify** from the drop-down menu.

      The **Modify DB Instance** window appears.

2. Type the maintenance window into the Maintenance Window text box using the format "day:hour:minute-day:hour:minute".

   Note
   The maintenance window and the backup window for the DB instance cannot overlap. If you enter a value for the maintenance window that overlaps the backup window, an error message appears.
3. Click the **OK** button.

   Changes to the maintenance window take effect immediately.

CLI

To adjust the preferred maintenance window

- Use the `rds-modify-db-instance` command with the following parameters:

```
PROMPT>rds-modify-db-instance mydbinstance --preferred-maintenance-window
Tue:04:00-Tue:04:30
```

This command produces output similar to the following.

```
DBINSTANCE  mydbinstance   2009-10-22T18:10:15.274Z  db.m1.large  mysql
  60
master  available   mydbinstance.clouwupjnvmq.us-east-1.rds.amazonaws.com
3306  us-east-1a  1  n  5.1.57  general-public-license
SECGROUP  default  active
PARAMGRP  default.mysql5.1  in-sync
```

API

To adjust the preferred maintenance window

- Call `ModifyDBInstance` with the following parameters:
  - *`DBInstanceIdentifier`* = `mydbinstance`
  - *`PreferredMaintenanceWindow`* = `Tue:04:00-Tue:04:30`

Example

```
https://rds.amazonaws.com/
 ?Action=ModifyDBInstance
 &DBInstanceIdentifier=mydbinstance
 &PreferredMaintenanceWindow=Tue:04:00-Tue:04:30
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-14T17%3A48%3A21.746Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Related Topics

- Amazon RDS Maintenance Window (p. 451)
- Amazon RDS DB Instances (p. 64)
- DB Instance Class (p. 65)

Modifying a DB Instance and Using the Apply Immediately Parameter

Most modifications to a DB instance can be applied immediately, applied during the next maintenance window, or manually applied when you reboot the instance. Some changes can result in an outage because Amazon RDS must reboot the instance for the change to take effect. Some modifications, such as changing a parameter group, require that you manually reboot the DB instance for the change to take effect. When you modify a DB instance, you have the option of applying the changes immediately by selecting the **Apply Immediately** option in the RDS console or setting the `ApplyImmediately` parameter to `true` using the CLI or RDS API.

The following table shows when a change is applied when you modify a DB instance setting, and the impact of selecting the **Apply Immediately** option in the RDS console or setting the `ApplyImmediately` parameter to `true` has on that change.

Note
Changing some DB instance settings cause an outage to occur when the DB instance is rebooted. Review the impact before changing settings.

DB Instance Setting	If Apply Immediately is set to true	If Apply Immediately is set to false
Backup retention period	Change is applied immediately. An immediate outage will occur if you change from 0 to a non-zero value or from a non-zero value to 0.	If you change the parameter from one non-zero value to another non-zero value, the change is asynchronously applied as soon as possible. In all other cases, the change is applied during the next maintenance window. An outage will occur if you change this parameter from 0 to a non-zero value or from a non-zero value to 0.
Automatically upgrade minor versions (applies only if you opted in to auto-upgrades when you created the DB instance)	No difference in when the change is applied. Change is asynchronously applied as soon as possible. An outage will occur if a newer minor version is available, and Amazon RDS has enabled auto patching for that engine version.	Change is asynchronously applied as soon as possible. An outage will occur if this parameter is set to true during the maintenance window, and a newer minor version is available, and RDS has enabled auto patching for that engine version.
Instance identifier	The name change is applied immediately and an immediate outage will occur.	The name change is applied during the next maintenance window. Changing this setting causes an outage to occur.
Instance class	Change is applied immediately and an immediate outage will occur.	Change is applied during the next maintenance window. Changing this setting causes an outage to occur.
Parameter group name	The name change is applied immediately. Any parameter value changes are applied to the DB instance after you manually reboot the DB instance.	The name change is applied immediately. Any parameter value changes are applied to the DB instance after you manually reboot the DB instance.

DB Instance Setting	If Apply Immediately is set to true	If Apply Immediately is set to false
Security group name	No difference in when the change is applied. Change is asynchronously applied as soon as possible.	Change is asynchronously applied as soon as possible.
Master password	No difference in when the change is applied. Change is asynchronously applied as soon as possible.	Change is asynchronously applied as soon as possible
Storage type	Change is applied immediately and in some cases, an immediate outage will occur. Changing from **Magnetic** to **General Purpose (SSD)** or **Provisioned IOPS (SSD)** will result in an outage. Also, changing from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic** will result in an outage. For more information about storage, see Storage for Amazon RDS (p. 77).	Change is applied during the next maintenance window and in some cases an immediate outage will occur at that time. Changing from **Magnetic** to **General Purpose (SSD)** or **Provisioned IOPS (SSD)** will result in an outage. Also, changing from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic** will result in an outage. For more information about storage, see Storage for Amazon RDS (p. 77).
Multi-AZ	Change is applied immediately.	Change is applied during the next maintenance window.
Option group	Change is applied immediately.	Change is applied during the next maintenance window. If the parameter change results in an option group that enables OEM, this change can cause a brief (sub-second) period during which new connections are rejected but existing connections are not interrupted.
Allocated storage	Change is made immediately. Performance may be degraded.	Change is applied during the next maintenance window. Performance may be degraded.
Preferred maintenance window	Change is applied immediately.	Change is applied immediately. If there are one or more pending actions that cause a outage, and the maintenance window is changed to include the current time, then those pending actions are applied immediately. If you set the window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure pending changes are applied.
Preferred backup window	No difference in when the change is applied. Change is asynchronously applied as soon as possible.	Change is asynchronously applied as soon as possible.

Renaming a DB Instance

You can rename a DB instance by using the AWS Management Console, the **rds-modify-db-instance** command, or the **ModifyDBInstance** API action. Renaming a DB instance can have far-reaching effects; the following is a list of things you should know before you rename a DB instance.

- When you rename a DB instance, the endpoint for the DB instance changes, because the URL includes the name you assigned to the DB instance. You should always redirect traffic from the old URL to the new one.
- When you rename a DB instance, the old DNS name that was used by the DB instance is immediately deleted, although it could remain cached for a few minutes. The new DNS name for the renamed DB instance becomes effective in about 10 minutes. The renamed DB instance is not available until the new name becomes effective.
- You cannot use an existing DB instance name when renaming an instance.
- All read replicas associated with a DB instance remain associated with that instance after it is renamed. For example, suppose you have a DB instance that serves your production database and the instance has several associated read replicas. If you rename the DB instance and then replace it in the production environment with a DB snapshot, the DB instance that you renamed will still have the read replicas associated with it.
- Metrics and events associated with the name of a DB instance will be maintained if you reuse a DB instance name. For example, if you promote a Read Replica and rename it to be the name of the previous master, the events and metrics associated with the master will be associated with the renamed instance.
- DB instance tags remain with the DB instance, regardless of renaming.
- DB snapshots are retained for a renamed DB instance.

The most common reasons for renaming a DB instance are that you are promoting a Read Replica or you are restoring data from a DB snapshot or PITR. By renaming the database, you can replace the DB instance without having to change any application code that references the DB instance. In these cases, you would do the following:

1. Stop all traffic going to the master DB instance. This can involve redirecting traffic from accessing the databases on the DB instance or some other way you want to use to prevent traffic from accessing your databases on the DB instance.
2. Rename the master DB instance to a name that indicates it is no longer the master as described later in this topic.
3. Create a new master DB instance by restoring from a DB snapshot or by promoting a read replica, and then give the new instance the name of the previous master DB instance.
4. Associate any read replicas with the new master DB instance.

If you delete the old master DB instance, you are responsible for deleting any unwanted DB snapshots of the old master instance. For information about promoting a Read Replica, see Working with PostgreSQL and MySQL Read Replicas (p. 472).

AWS Management Console

To rename a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, select **DB Instances**.
3. Select the check box next to the DB instance you want to rename.

4. From the **Instance Actions** dropdown menu, select **Modify**.
5. Enter a new name in the **DB Instance Identifier** text box. Select the **Apply Immediately** check box, and then click **Continue**.
6. Click **Modify DB Instance** to complete the change.

CLI

To rename a DB instance

- Use the command `rds-modify-db-instance` and provide the old *`DBInstanceIdentifier`* value and use the –n switch with the name of the new DB instance. The syntax is as follows:

```
PROMPT>rds-modify-db-instance DBInstanceIdentifier -n NewDBInstanceIdentifier
```

API

To rename a DB instance

- Call `ModifyDBInstance` with the following parameters:
  - *`NewDBInstanceIdentifier`* = new name for the instance

Related Topics

- Promoting a Read Replica to Be a DB Instance (p. 477)
- Amazon RDS DB Instances (p. 64)

Deleting a DB Instance

You can delete a DB instance in any state and at any time. To delete a DB instance, you must specify the name of the instance and specify if you want to have a final DB snapshot taken of the instance. If the DB instance you are deleting has a status of "Creating," you will not be able to have a final DB snapshot taken. If the DB instance is in a failure state with a status of "failed," "incompatible-restore," or "incompatible-network," you can only delete the instance when the `SkipFinalSnapshot` parameter is set to "`true`."

> **Important**
> If you choose not to create a final DB snapshot, you will not be able to later restore the DB instance to its final state. When you delete a DB instance, all automated backups are deleted and cannot be recovered. Manual DB snapshots of the instance are not deleted.

If the DB instance you want to delete has a Read Replica, you should either promote the Read Replica or delete it. For more information on promoting a Read Replica, see Promoting a Read Replica to Be a DB Instance (p. 477)

In the following examples, you delete a DB instance both with and without a final DB snapshot.

Deleting a DB Instance with No Final Snapshot

You can skip creating a final DB snapshot if you want to quickly delete a DB instance. Note that when you delete a DB instance, all automated backups are deleted and cannot be recovered. Manual snapshots are not deleted.

AWS Management Console

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the **DB Instances** list, select the check box next to the DB instance you wish to delete.
3. Click **Instance Actions**, and then select **Delete** from the context menu.
4. Select **No** in the **Create final Snapshot?** drop-down list box.
5. Click **Yes, Delete**.

CLI

To delete a DB instance with no final DB snapshot

- Use the command `rds-delete-db-instance` to delete an instance.

```
PROMPT>rds-delete-db-instance mydbinstance mydbinstance --skip-final-snapshot
```

API

To delete a DB instance with no final DB snapshot

- Call `DeleteDBInstance` with the following parameters:
  - *`DBInstanceIdentifier`* = `mydbinstance`

- *SkipFinalSnapshot* = true

Example

```
https://rds.amazonaws.com/
 ?Action=DeleteDBInstance
 &DBInstanceIdentifier=mydbinstance
 &SkipFinalSnapshot=true
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-14T22%3A20%3A46.297Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Deleting a DB Instance with a Final Snapshot

You can create a final DB snapshot if you want to be able to restore a deleted DB instance at a later time. All automated backups will also be deleted and cannot be recovered. Manual snapshots are not deleted.

AWS Management Console

To delete a DB instance with a final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the **DB Instances** list, select the check box next to the DB Instance you wish to delete.
3. Click **Instance Actions**, and then select **Delete** from the context menu.
4. Select **Yes** in the **Create final Snapshot?** drop-down list box.
5. Type the name of your final DB snapshot into the **Final Snapshot name** text box.
6. Click **Yes, Delete**.

CLI

To delete a DB instance with a final DB snapshot

- Use the command `rds-delete-db-instance` to delete an instance.

```
PROMPT>rds-delete-db-instance mydbinstance mydbinstance --final-snapshot-
identifier myfinaldbsnapshot
```

This command should produce output similar to the following:

```
Once you begin deleting this database, it will no longer be able to accept
connections.
Are you sure you want to delete this database? [Ny]y
DBINSTANCE  mydbinstance  2009-10-21T01:54:49.521Z  db.m1.large  MySQL  50
```

```
   sa  deleting  us-east-1a  3
   SECGROUP  default  active
```

API

To delete a DB instance with a final DB snapshot

- Call `DeleteDBInstance` with the following parameters:
  - *`DBInstanceIdentifier`* = `mydbinstance`
  - *`FinalDBSnapshotIdentifier`* = `myfinaldbsnapshot`

Example

```
https://rds.amazonaws.com/
 ?Action=DeleteDBInstance
 &DBInstanceIdentifier=mydbinstance
 &FinalDBSnapshotIdentifier=myfinaldbsnapshot
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-14T22%3A20%3A46.297Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Related Topics

- Creating a DB Instance Running the MySQL Database Engine (p. 129)
- Amazon RDS DB Instances (p. 64)

Rebooting a DB Instance

In some cases, if you modify a DB instance, the DB parameter group associated with the instance, or a static DB parameter in a parameter group the instances uses, you must reboot the instance for the changes to take effect.

Rebooting a DB instance restarts the database engine service. A reboot also applies to the DB instance any modifications to the associated DB parameter group that were pending. Rebooting a DB instance results in a momentary outage of the instance, during which the DB instance status is set to *rebooting*. If the Amazon RDS instance is configured for MultiAZ, it is possible that the reboot will be conducted through a failover. An Amazon RDS event is created when the reboot is completed.

If your DB instance is deployed in multiple Availability Zones, you can force a failover from one AZ to the other when you select the **Reboot** option. When you force a failover of your DB instance, Amazon RDS automatically switches to a standby replica in another Availability Zone and updates the DNS record for the DB instance to point to the standby DB instance. As a result, you will need to clean up and re-establish any existing connections to your DB instance. **Reboot with failover** is beneficial when you want to simulate a failure of a DB instance for testing, or restore operations to the original AZ after a failover occurs. For more information, see High Availability (Multi-AZ).

The time required to reboot is a function of the specific database engine's crash recovery process. To improve the reboot time, we recommend that you reduce database activities as much as possible during the reboot process to reduce rollback activity for in-transit transactions.

AWS Management Console

To reboot a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Instances**.
3. Select the check box of the DB instance that you want to reboot.
4. Select **Instance Actions** and then select **Reboot** from the drop down menu.
5. To force a failover from one AZ to another, select the **Reboot with failover?** check box in the **Reboot DB Instance** dialog box.
6. Click **Yes, Reboot**. To cancel the reboot instead, click **Cancel**.

CLI

To reboot a DB instance

- Use the `rds-reboot-db-instance` command. To force a failover from one AZ to the other, use the `force-failover` parameter.

```
PROMPT>rds-reboot-db-instance dbInstanceID --force-failover true
```

API

To reboot a DB instance

- Call `RebootDBInstance`. To force a failover from one AZ to the other, add the following parameter:
  - `ForceFailover = true`

Working with Storage Types

Data storage in Amazon RDS is specified by selecting a storage type and providing a storage size (GB) when you create or modify a DB instance. You can change the type of storage your instance uses by modifying the DB instance, but changing the type of storage in some cases might result in a short outage for the instance. Changing from Magnetic to either General Purpose (SSD) or Provisioned IOPS (SSD) will result in an outage. Also, changing from General Purpose (SSD) or Provisioned IOPS (SSD) to Magnetic will result in an outage. The outage time is typically 60-120 seconds. For more information about Amazon RDS storage types, see Amazon RDS Storage Types (p. 77).

Increasing the allocated storage will not result in an outage. Note that you cannot reduce the amount of storage once it has been allocated. The only way to reduce the amount of storage allocated to a DB instance is to dump the data out of the DB instance, create a new DB instance with less storage space, and then load the data into the new DB instance.

When estimating your storage needs, take into consideration that Amazon RDS allocates a minimum amount of storage for file system structures. This reserved space can be up to 3% of the allocated storage for a DB instance, though in most cases the reserved space is far less. You should set up a CloudWatch alarm for your DB instance's free storage space and react when necessary. For information on setting CloudWatch alarms, see the CloudWatch Getting Started Guide.

Topics

- Modifying a DB Instance to Use a Different Storage Type (p. 464)
- Modifying IOPS and Storage Settings for a DB Instance That Uses Provisioned IOPS Storage (p. 466)
- Creating a DB Instance that Uses Provisioned IOPS Storage (p. 468)
- Creating a MySQL Read Replica That Uses Provisioned IOPS Storage (p. 470)

Modifying a DB Instance to Use a Different Storage Type

You can use the Amazon RDS console, the Amazon RDS API, or the Command Line Interface (CLI) to modify a DB instance to use Standard, General Purpose (SSD), or Provisioned IOPS storage. You must specify either a value for allocated storage or specify both allocated storage and IOPS values. You may need to modify the amount of allocated storage in order to maintain the required ratio between IOPS and storage. For more information about the required ratio between IOPS and storage, see the Using Provisioned IOPS Storage with Multi-AZ, Read Replicas, Snapshots, VPC, and DB Instance Classes (p. 83).

> **Note**
> You cannot modify an existing SQL Server DB instance to change storage type or modify storage allocation.

In some cases an immediate outage will occur when you convert from one storage type to another. If you change from **Magnetic** to **General Purpose (SSD)** or **Provisioned IOPS (SSD)**, a short outage will occur. Also, if you change from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic**, a short outage will occur. For DB instances in a single Availability Zone, the DB instance could be unavailable for a few minutes when the conversion is initiated. For multi-AZ deployments, the time the DB instance is unavailable is limited to the time it takes for a failover operation to complete, which typically takes less than two minutes. Although your DB instance is available for reads and writes during the conversion, you may experience degraded performance until the conversion process is complete. This process can take several hours.

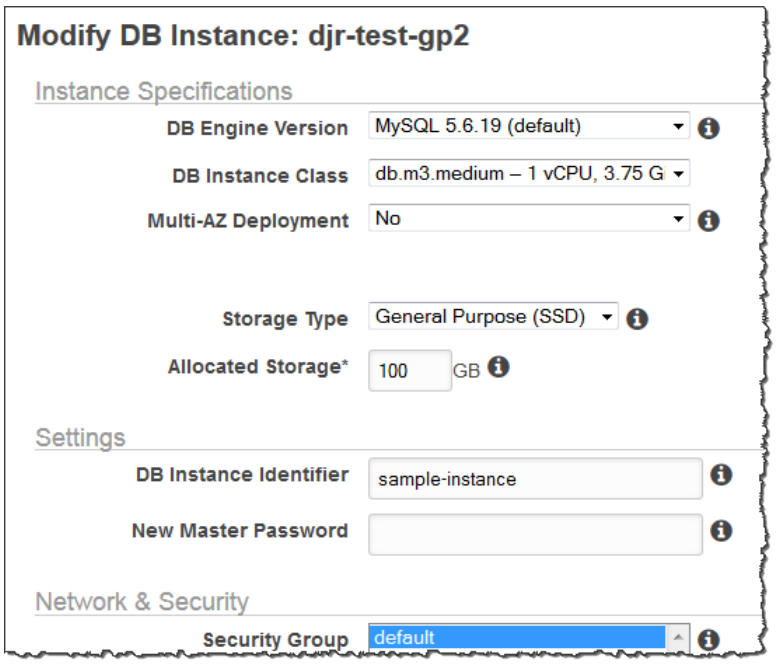
Whenever you change the storage type of a DB instance, the data for that DB instance is migrated to a new volume. The duration of the migration depends on several factors such as database load, storage

size, storage type, amount of IOPS provisioned (if any). Typical migration times will be under 24 hours, but can take up to several days in some cases. During the migration, the DB instance will be available for use, but may experience performance degradation. **While the migration takes place, nightly backups will be suspended and no other Amazon RDS operations can take place, including Modify, Reboot, Delete, Create Read Replica, and Take DB Snapshot**.

AWS Management Console

To modify a DB instance to use a different storage type

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. From the navigation pane on the Amazon RDS Console, click **DB Instances**.
3. Select the DB instance that you want to modify.
4. In the **Instance Actions** dropdown list, click **Modify**.
5. Select the new **Storage Type** for the DB instance and enter a value for **Allocated Storage**. If you are modifying your DB instance to use the Provisioned IOPS storage type, then you must also provide a **Provisioned IOPS** value. For more information, go to Modifying IOPS and Storage Settings for a DB Instance That Uses Provisioned IOPS Storage (p. 466).



6. To immediately initiate conversion of the DB instance to use the new storage type, select the **Apply Immediately** check box. If the check box is cleared (the default), the changes will be applied during the next maintenance window. In some cases an immediate outage will occur when the conversion is applied. Changing from **Magnetic** to **General Purpose (SSD)** or **Provisioned IOPS (SSD)** will result in an outage. Also, changing from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic** will result in an outage. For more information about storage, see Storage for Amazon RDS (p. 77).
7. When the settings are as you want them, click **Continue**.

CLI

To modify a DB instance to use a different storage type

Use the **rds-modify-db-instance** command. Set the following parameters:

- `--allocated-storage` – Amount of storage to be allocated for the DB instance, in gigabytes.
- `--storage-type` – The new storage type for the DB instance. You can specify `gp2` for general purpose (SSD), `io1` for Provisioned IOPS), or `standard` for magnetic storage.
- `--apply-immediately` – Set to `True` to initiate conversion immediately. If `False` (the default), the conversion is applied during the next maintenance window. In some cases an immediate outage will occur when the conversion is applied. Changing from **Magnetic** to **General Purpose (SSD)** or **Provisioned IOPS (SSD)** will result in an outage. Also, changing from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic** will result in an outage. For more information about storage, see Storage for Amazon RDS (p. 77).

API

Use the **ModifyDBInstance** action. Set the following parameters:

- `AllocatedStorage` – Amount of storage to be allocated for the DB instance, in gigabytes.
- `StorageType` – The new storage type for the DB instance. You can specify `gp2` for general purpose (SSD), `io1` for Provisioned IOPS), or `standard` for magnetic storage.
- `ApplyImmediately` – Set to `True` if you want to initiate conversion immediately. If `False` (the default), the conversion is applied during the next maintenance window. In some cases an immediate outage will occur when the conversion is applied. Changing from **Magnetic** to **General Purpose (SSD)** or **Provisioned IOPS (SSD)** will result in an outage. Also, changing from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic** will result in an outage. For more information about storage, see Storage for Amazon RDS (p. 77).

Modifying IOPS and Storage Settings for a DB Instance That Uses Provisioned IOPS Storage

You can modify the settings for an Oracle, PostgreSQL, or MySQL DB instance that uses Provisioned IOPS storage by using the AWS Management Console, the Amazon RDS API, or the Command Line Interface (CLI). You must specify the storage type, allocated storage, and the amount of Provisioned IOPS that you require. You can choose from 1000 IOPS and 100GB of storage up to 30,000 IOPS and 3 TB (3000 GB) of storage, depending on your database engine. You cannot reduce the amount of allocated storage from the value currently allocated for the DB instance. For more information, see the Using Provisioned IOPS Storage with Multi-AZ, Read Replicas, Snapshots, VPC, and DB Instance Classes (p. 83).

> **Note**
> You cannot modify the IOPS rate or allocated storage settings for a SQL Server DB instance.

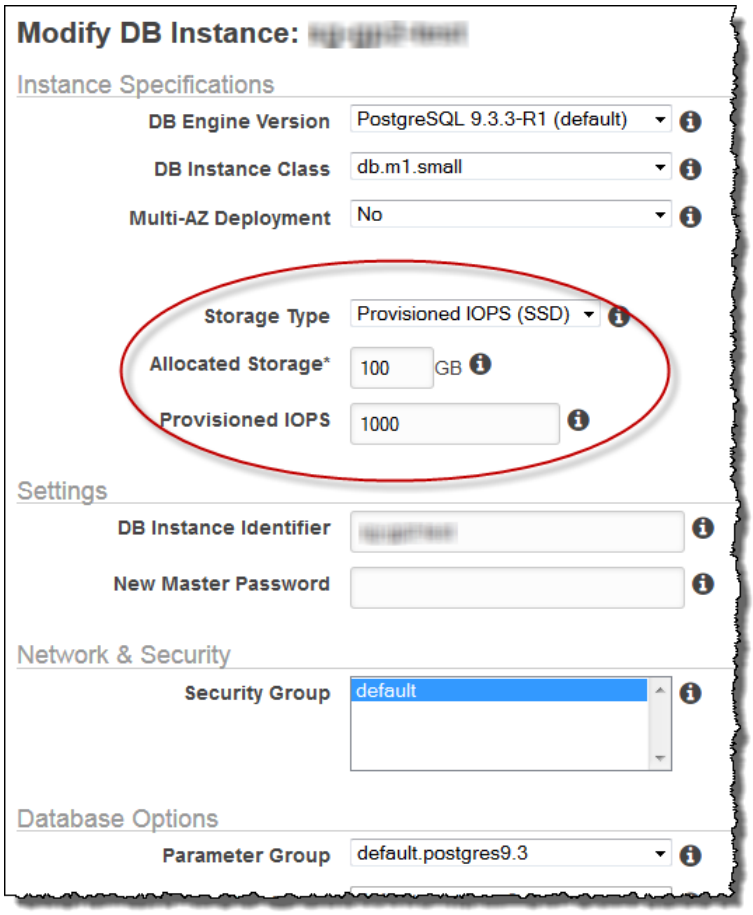
AWS Management Console

To modify the Provisioned IOPS settings for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **DB Instances**.

Note
To filter the list of DB instances, in the **Search DB Instances...** box, type a text string that Amazon RDS will use to filter the results. Only DB instances whose name contains the string will appear.

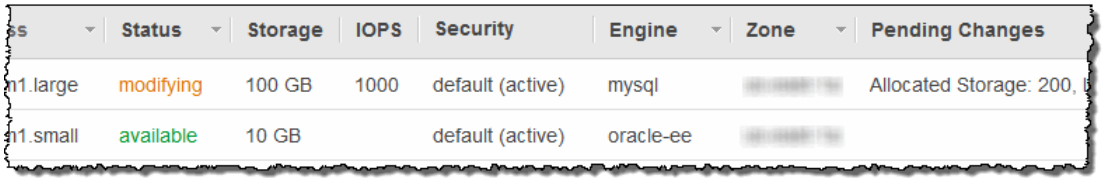
3. Select the DB instance with Provisioned IOPS storage that you want to modify.
4. In the **Instance Actions** dropdown list, click **Modify**.
5. On the **Modify DB Instance** page, enter the value that you want for either **Allocated Storage** or **Provisioned IOPS**.



If the value you specify for either **Allocated Storage** or **Provisioned IOPS** is outside the limits supported by the other parameter, a warning message is displayed indicating the range of values required for the other parameter.

6. To apply the changes to the DB instance immediately, select the **Apply Immediately** check box. If you leave the check box cleared, the changes will be applied during the next maintenance window.
7. Click **Continue**.
8. Review the parameters that will be changed, and click **Modify DB Instance** to complete the modification.

   The new value for allocated storage or for provisioned IOPS appears in the **Pending Values** column.



ss	Status	Storage	IOPS	Security	Engine	Zone	Pending Changes
m1.large	modifying	100 GB	1000	default (active)	mysql	[redacted]	Allocated Storage: 200, I
m1.small	available	10 GB		default (active)	oracle-ee	[redacted]	

CLI

To modify the Provisioned IOPS settings for a DB instance

Use the **rds-modify-db-instance** command. Set the following parameters:

- `--storage-type` – Set to `io1` for Provisioned IOPS.
- `--allocated-storage` – Amount of storage to be allocated for the DB instance, in gigabytes.
- `--iops` – The new amount of Provisioned IOPS for the DB instance, expressed in I/O operations per second.
- `--apply-immediately` – Set to `True` to initiate conversion immediately. If `False` (the default), the conversion is applied during the next maintenance window.

API

To modify the Provisioned IOPS settings for a DB instance

Use the **ModifyDBInstance** action. Set the following parameters:

- `StorageType` – Set to `io1` for Provisioned IOPS.
- `AllocatedStorage` – Amount of storage to be allocated for the DB instance, in gigabytes.
- `Iops` – The new IOPS rate for the DB instance, expressed in I/O operations per second.
- `ApplyImmediately` – Set to `True` if you want to initiate conversion immediately. If `False` (the default), the conversion will be applied during the next maintenance window.

Creating a DB Instance that Uses Provisioned IOPS Storage

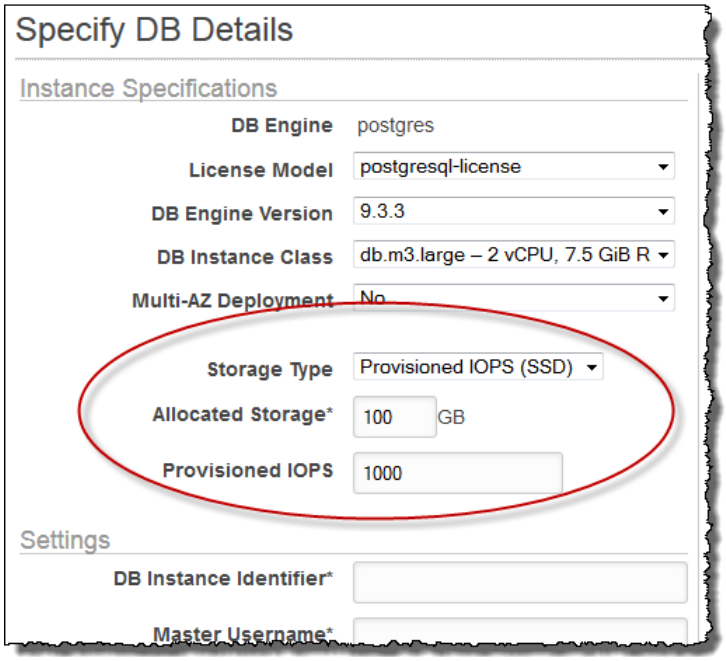
You can create a DB instance that uses Provisioned IOPS by setting several parameters when you launch the DB instance. You can use the AWS Management Console, the Amazon RDS API, or the Command Line Interface (CLI). For more information about the settings you should use when creating a DB instance, see Creating a DB Instance Running the MySQL Database Engine (p. 129), Creating a DB Instance Running the Oracle Database Engine (p. 205), or Creating a DB Instance Running the SQL Server Database Engine (p. 313).

AWS Management Console

To create a new DB instance that uses Provisioned IOPS storage

For a complete description on how to create a DB instance, see the topic for your DB engine: Creating a DB Instance Running the MySQL Database Engine (p. 129), Creating a DB Instance Running the Oracle Database Engine (p. 205), Creating a DB Instance Running the PostgreSQL Database Engine (p. 363), Creating a DB Instance Running the SQL Server Database Engine (p. 313)

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. From the Amazon RDS Console, click **Launch DB Instance**.
3. In the **Launch RDS DB Instance Wizard**, on the **Engine Selection** page, click the **Select** button next to the DB engine that you want.
4. On the **Specify DB Details** page, select **`Provisioned IOPS (SSD)`** for **Storage Type**.
5. Specify values for **Allocated Storage** and **Provisioned IOPS**. You can change these values but the ratio between provisioned IOPS and allocated storage must be in a range between 3:1 and 10:1 for MySQL and Oracle instances, while SQL Server requires a ratio of 10:1.



6. When the settings are as you want them, click **Continue**. Enter the remaining values to create the DB instance.

CLI

To create a new DB instance that uses Provisioned IOPS storage

Use the **rds-create-db-instance** command. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- `--storage-type` – Set to `io1` for Provisioned IOPS.
- `--allocated-storage` - Amount of storage to be allocated for the DB instance, in gigabytes.
- `--iops` - The new IOPS rate for the DB instance, expressed in I/O operations per second.

API

To create a new DB instance that uses Provisioned IOPS storage

Use the `CreateDBInstance` action. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- `StorageType` – Set to `io1` for Provisioned IOPS.
- `AllocatedStorage` - Amount of storage to be allocated for the DB instance, in gigabytes.
- `Iops` - The new IOPS rate for the DB instance, expressed in I/O operations per second.

Creating a MySQL Read Replica That Uses Provisioned IOPS Storage

You can create a MySQL Read Replica that uses Provisioned IOPS storage. You can create a Read Replica that uses Provisioned IOPS storage by using a source DB instance that uses either standard storage or Provisioned IOPS storage.

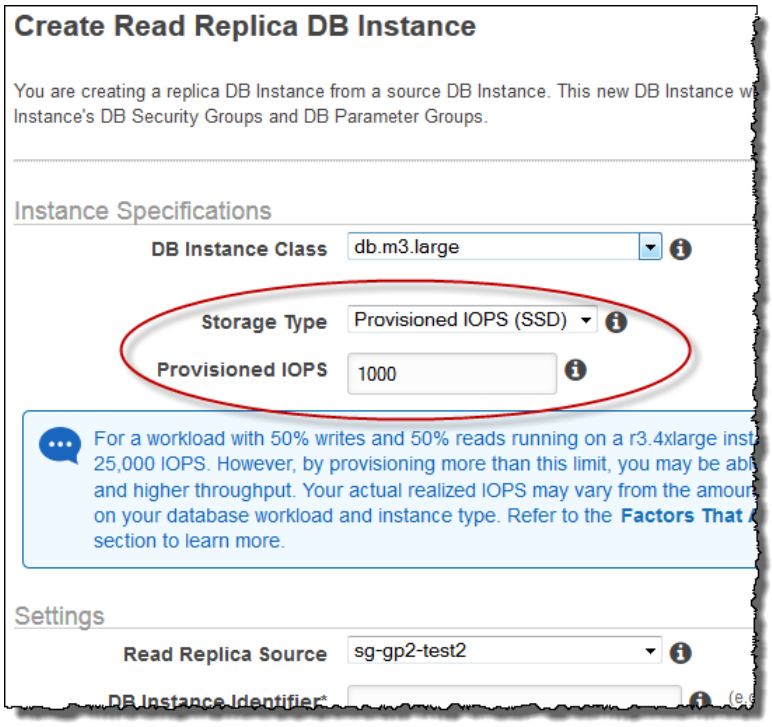
AWS Management Console

To create a Read Replica DB instance that uses Provisioned IOPS storage

For a complete description on how to create a Read Replica, see Creating a Read Replica (p. 476)

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the **Navigation** pane, click **DB Instances**.
3. Select the MySQL DB instance with Provisioned IOPS storage that you want to use as the source for the Read Replica, and then click **Instance Actions** and select **Create Read Replica**.

   Important
   The MySQL DB instance that your are creating a Read Replica for must have allocated storage within the range of storage for MySQL PIOPS (100 GB - 3 TB). If the allocated storage for the source MySQL DB instance is not within the range of storage for MySQL, then the **Provisioned IOPS** storage type will not be available as an option when creating the Read Replica. Instead, only the **GP2** or **Standard** storage types can be set. You can modify the allocated storage for the source MySQL DB instance to be within the range of storage for MySQL PIOPS before creating a Read Replica. For more information on PIOPS range of storage, see Amazon RDS Provisioned IOPS Storage to Improve Performance (p. 82). For information on modifying a DB instance, see Modifying a DB Instance Running the MySQL Database Engine (p. 140).
4. On the **Create Read Replica DB Instance** page, type a DB instance identifier for the Read Replica.


Create Read Replica DB Instance
You are creating a replica DB Instance from a source DB Instance. This new DB Instance w
Instance's DB Security Groups and DB Parameter Groups.
Instance Specifications
DB Instance Class
db.m3.large
Storage Type
Provisioned IOPS (SSD)
Provisioned IOPS
1000
For a workload with 50% writes and 50% reads running on a r3.4xlarge inst
25,000 IOPS. However, by provisioning more than this limit, you may be abl
and higher throughput. Your actual realized IOPS may vary from the amoun
on your database workload and instance type. Refer to the Factors That
section to learn more.
Settings
Read Replica Source
sg-gp2-test2
DB Instance Identifier*

5. Click **Yes, Create Read Replica**.

CLI

To create a Read Replica DB instance that uses Provisioned IOPS

Use the **rds-create-db-instance-read-replica** command. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- `--allocated-storage` - Amount of storage to be allocated for the DB instance, in gigabytes.
- `--iops` - The new IOPS rate for the DB instance, expressed in I/O operations per second.

API

To create a Read Replica DB instance that uses Provisioned IOPS

Use the `CreateDBInstanceReadReplica` action. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- `AllocatedStorage` - Amount of storage to be allocated for the DB instance, in gigabytes.
- `Iops` - The new IOPS rate for the DB instance, expressed in I/O operations per second.

Working with PostgreSQL and MySQL Read Replicas

Amazon RDS uses the MySQL and PostgreSQL (version 9.3.5 and later) DB engines' built-in replication functionality to create a special type of DB instance called a Read Replica from a source DB instance. Updates made to the source DB instance are asynchronously copied to the Read Replica. You can reduce the load on your source DB instance by routing read queries from your applications to the Read Replica. Read Replicas allow you to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.

Note that the information in this topic applies to creating Amazon RDS Read Replicas, either in the same region as the source DB instance, or in a separate region for MySQL Read Replicas. This topic does not apply to setting up replication with an instance that is running on an Amazon EC2 instance or that is on-premises.

When you create a Read Replica, you first specify an existing DB instance as the source. Then, Amazon RDS takes a snapshot of the source instance and creates a read-only instance from the snapshot. Amazon RDS then uses the asynchronous replication method for the DB engine to update the Read Replica whenever there is a change to the source DB instance. The Read Replica operates as a DB instance that allows only read-only connections; applications can connect to a Read Replica the same way they would to any DB instance. Amazon RDS replicates all databases in the source DB instance.

Amazon RDS sets up a secure communications channel between the source DB instance and a Read Replica. Amazon RDS establishes any AWS security configurations, such as adding security group entries, needed to enable the secure channel. MySQL DB instances use public key encryption between the source DB instance and the Read Replica. PostgreSQL DB instances use a secure connection that you can encrypt by setting the *ssl* parameter to **1** for both the source and the replica instances.

Topics

- Amazon RDS Read Replica Overview (p. 472)
- PostgreSQL Read Replicas (version 9.3.5 and later) (p. 473)
- MySQL Read Replicas (p. 475)
- Creating a Read Replica (p. 476)
- Promoting a Read Replica to Be a DB Instance (p. 477)
- Replicating a Read Replica Across Regions (MySQL only) (p. 479)
- Monitoring Read Replication (p. 481)
- Troubleshooting a MySQL Read Replica Problem (p. 483)
- Troubleshooting a PostgreSQL Read Replica Problem (p. 484)

Amazon RDS Read Replica Overview

There are a variety of scenarios where deploying one or more Read Replica for a given source DB instance might make sense. Common reasons for deploying a Read Replica include the following:

- Scaling beyond the compute or I/O capacity of a single DB instance for read-heavy database workloads. This excess read traffic can be directed to one or more Read Replicas.
- Serving read traffic while the source DB instance is unavailable. If your source DB instance cannot take I/O requests (for example, due to I/O suspension for backups or scheduled maintenance), you can direct read traffic to your Read Replica(s). For this use case, keep in mind that the data on the Read Replica might be "stale" because the source DB instance is unavailable.
- Business reporting or data warehousing scenarios where you might want business reporting queries to run against a Read Replica, rather than your primary, production DB instance.

By default, a Read Replica is created with the same storage type as the source DB instance. However, you can create a Read Replica that has a different storage type from the source DB instance based on the options listed in the following table.

Source DB Instance Storage Type	Source DB Instance Storage Allocation	Read Replica Storage Type Options
PIOPS	100 GB - 3 TB	PIOPS | GP2 | Standard
GP2	100 GB - 3 TB	PIOPS | GP2 | Standard
GP2	Less than 100 GB	GP2 | Standard
Standard	100 GB - 3 TB	PIOPS | GP2 | Standard
Standard	Less than 100 GB	GP2 | Standard

Amazon RDS does not support circular replication. You cannot configure a DB instance to serve as a replication source to an existing DB instance; you can only create a new Read Replica from an existing DB instance. For example, if MyDBInstance replicates to ReadReplica1, you cannot configure ReadReplica1 to replicate back to MyDBInstance. From ReadReplica1, you can only create a new Read Replica, such as ReadReplica2.

Differences Between PostgreSQL and MySQL Read Replicas

Because the PostgreSQL and MySQL DB engines implement replication differently, they have several significant differences you should know about:

Feature/Behavior	PostgreSQL	MySQL
What is the replication method?	Physical replication.	Logical replication.
How are transaction logs purged?	PostgreSQL has a parameter, `wal_keep_segments`, that dictates how many Write Ahead Log (WAL) files are kept to provide data to the Read Replicas. The parameter value species the number of logs to keep.	Amazon RDS won't delete any binary logs that have not been applied.
Can a replica be made writable?	No. A PostgreSQL Read Replica is a physical copy and PostgreSQL doesn't allow for a Read Replica to be made writeable.	Yes. You can enable the MySQL Read Replica to be writable.
Can backups be performed on the replica?	Yes, you can create a snapshot of a PostgreSQL Read Replica, but you cannot enable automatic backups.	Yes. You can enable automatic backups on a MySQL Read Replica.
Can you use parallel replication?	No. PostgreSQL has a single process handling replication.	Yes. MySQL version 5.6 allows for parallel replication threads.

PostgreSQL Read Replicas (version 9.3.5 and later)

Amazon RDS PostgreSQL 9.3.5 and later uses PostgreSQL native streaming replication to create a read-only copy of a source (a "master" in Postgres terms) DB instance. This Read Replica (a "standby"

in Postgres terms) DB instance is an asynchronously created physical replication of the master DB instance. It is created by a special connection that transmits WAL data between the source DB instance and the Read Replica where PostgreSQL asynchronously streams database changes as they are made.

PostgreSQL uses a "replication" role to perform streaming replication. The role is privileged, but, can not be used to modify any data. PostgreSQL uses a single process for handling replication.

Creating a PostgreSQL Read Replica does not require an outage for the master DB instance. Amazon RDS sets the necessary parameters and permissions for the source DB instance and the Read Replica without any service interruption. A snapshot is taken of the source DB instance and this snapshot becomes the Read Replica. No outage occurs when you delete a Read Replica either.

You can create up to five Read Replicas from one source DB instance. For replication to operate effectively, each Read Replica should have the same amount of compute and storage resources as the source DB instance. If you scale the source DB instance, you should also scale the Read Replicas.

Amazon RDS will override any incompatible parameters on the Read Replica if it prevents the Read Replica from starting. For example, if the `max_connections` parameter value is higher on the source DB instance than on the Read Replica, Amazon RDS will update the parameter on the Read Replica to be the same value as that on the source DB instance.

Here are some important facts about PostgreSQL Read Replicas:

- You can create PostgreSQL Read Replicas only in the same region as the source DB instance.
- PostgreSQL Read Replicas are read-only and cannot be made writeable.
- You cannot create a Read Replica from another Read Replica (that is, you cannot create cascading Read Replicas).
- You can promote a PostgreSQL Read Replica to be a new source DB instance. Note that the Read Replica does not become the new source DB instance automatically. The Read Replica, when promoted, stops receiving WAL communications and is no longer a read-only instance. You must set up any replication you intend going forward because the promoted Read Replica is now a new source DB instance.
- A PostgreSQL Read Replica will report a replication lag of up to five minutes if there are no user transactions occurring on the source DB instance.
- Before a DB instance can serve as a source DB instance, you must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0.

Situations That Break PostgreSQL Replication

There are several situations where a PostgreSQL source DB instance can unintentionally break replication with a Read Replica. These situations include the following:

- The `max_wal_senders` parameter is set too low to provide enough data to the number of Read Replicas. This situation causes replication to stop.
- The PostgreSQL parameter, `wal_keep_segments`, dictates how many Write Ahead Log (WAL) files are kept to provide data to the Read Replicas. The parameter value species the number of logs to keep. If you set the parameter value too low, you can cause a Read Replica to fall so far behind that streaming replication stops. In this case, Amazon RDS will report a replication error and begin recovery on the Read Replica by replaying the source DB instance's archived WAL logs. This recovery process continues until the Read Replica has caught up enough to continue streaming replication. For more information on this process and how to determine the appropriate parameter setting, see Troubleshooting a PostgreSQL Read Replica Problem (p. 484).
- A PostgreSQL Read Replica will require a reboot if the source DB instance endpoint changes.

When the WAL stream that provides data to a Read Replica is broken, PostgreSQL switches into recovery mode to restore the Read Replica by using archived WAL files. Once this process is compete, PostgreSQL will attempt to re-establish streaming replication.

MySQL Read Replicas

Before a MySQL DB instance can serve as a replication source, you must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0. This requirement also applies to a Read Replica that is the source DB instance for another Read Replica. Automatic backups are supported only for Read Replicas running MySQL 5.6, not 5.1 or 5.5.

You can create up to five Read Replicas from one DB instance. In order for replication to operate effectively, each Read Replica should have as much compute and storage resources as the source DB instance. If you scale the source DB instance, you should also scale the Read Replicas.

If a Read Replica is running MySQL 5.6, you can specify it as the source DB instance for another Read Replica. For example, you can create ReadReplica1 from MyDBInstance, and then create ReadReplica2 from ReadReplica1. Updates made to MyDBInstance are replicated to ReadReplica1 and then replicated from ReadReplica1 to ReadReplica2. You cannot have more than three instances involved in a replication chain. For example, you can create ReadReplica1 from MySourceDBInstance, and then create ReadReplica2 from ReadReplica1, but you cannot create a ReadReplica3 from ReadReplica2. To enable automatic backups on an Amazon RDS MySQL version 5.6 Read Replica, first create the Read Replica, then modify the Read Replica to enable automatic backups.

Read Replicas are designed to support read queries, but there may be a need for occasional updates, such as adding an index to speed the specific types of queries accessing the replica. You can enable updates by setting the **read_only** parameter to **0** in the DB parameter group for the Read Replica.

You can run multiple concurrent Read Replica create or delete actions that reference the same source DB instance, as long as you stay within the limit of five Read Replicas for the source instance.

You can create a Read Replica from either Single-AZ or Multi-AZ DB instance deployments. You use a Multi-AZ deployment to improve the durability and availability of a critical system, but you cannot use the Multi-AZ secondary to serve read-only queries. You must create Read Replicas from a high-traffic, Multi-AZ DB instance to offload read queries from the source DB instance. If the source instance of a Multi-AZ deployment fails over to the secondary, any associated Read Replicas will be switched to use the secondary as their replication source. It is possible that the Read Replicas cannot be switched to the secondary if some MySQL binlog events are not flushed during the failure. In this case, you must manually delete and recreate the Read Replicas. You can reduce the chance of this happening in MySQL 5.1 or 5.5 by setting the **sync_binlog=1** and **innodb_support_xa=1** dynamic variables. These settings may reduce performance, so test their impact before implementing the changes to a production environment. These problems are less likely to occur if you are using MySQL 5.6. For instances running MySQL 5.6, the parameters are set by default to **sync_binlog=1** and **innodb_support_xa=1**.

You usually configure replication between Amazon RDS DB instances, but you can configure replication to import databases from instances of MySQL running outside of Amazon RDS, or to export databases to such instances. For more information, see Importing Data to an Amazon RDS MySQL DB Instance with Reduced Downtime (p. 147) and Using Replication to Export MySQL 5.6 Data (p. 164).

You can stop and restart the replication process on an Amazon RDS DB instance by calling the system stored procedures mysql.rds_stop_replication (p. 181) and mysql.rds_start_replication (p. 180). You can do this when replicating between two Amazon RDS instances for long running operations such as creating large indexes. You also need to stop and start replication when importing or exporting databases. For more information, see Importing Data to an Amazon RDS MySQL DB Instance with Reduced Downtime (p. 147) and Using Replication to Export MySQL 5.6 Data (p. 164).

You must explicitly delete Read Replicas, using the same mechanisms for deleting a DB instance. If you delete the source DB instance without deleting the replicas, each replica is promoted to a standalone, Single-AZ DB instance.

If you promote a MySQL Read Replica that is in turn replicating to other Read Replicas, those replications stay active. Consider an example where MyDBInstance1 replicates to MyDBInstance2, and MyDBInstance2 replicates to MyDBInstance3. If you promote MyDBInstance2, there will no longer be any replication from MyDBInstance1 to MyDBInstance2, but MyDBInstance2 will still replicate to MyDBInstance3.

Creating a Read Replica

You can create a Read Replica from an existing MySQL or PostgreSQL DB instance using the AWS Management Console, CLI, or API. You create a Read Replica by specifying the `SourceDBInstanceIdentifier`, which is the DB instance identifier of the source DB instance from which you wish to replicate.

When you initiate the creation of a Read Replica, Amazon RDS takes a DB snapshot of your source DB instance and begins replication. As a result, you will experience a brief I/O suspension on your source DB instance as the DB snapshot occurs. The I/O suspension typically lasts about one minute and can be avoided if the source DB instance is a Multi-AZ deployment (in the case of Multi-AZ deployments, DB snapshots are taken from the standby). An active, long-running transaction can slow the process of creating the Read Replica, so wait for long-running transactions to complete before creating a Read Replica. If you create multiple Read Replicas in parallel from the same source DB instance, Amazon RDS takes only one snapshot at the start of the first create action.

When creating a Read Replica, there are a few things to consider. First, you must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0. This requirement also applies to a Read Replica that is the source DB instance for another Read Replica. For MySQL DB instances, automatic backups are supported only for Read Replicas running MySQL 5.6 but not for MySQL versions 5.1 or 5.5. To enable automatic backups on an Amazon RDS MySQL version 5.6 Read Replica, first create the Read Replica, then modify the Read Replica to enable automatic backups.

Preparing MySQL DB Instances That Use MyISAM

If your MySQL DB instance uses a non-transactional engine such as MyISAM, you will need to perform the following steps to successfully set up your Read Replica. These steps are required to ensure that the Read Replica has a consistent copy of your data. Note that these steps are not required if all of your tables use a transactional engine such as InnoDB.

1. Stop all DML and DDL operations on non-transactional tables in the source DB instance and wait for them to complete. SELECT statements can continue running.
2. Flush and lock the tables in the source DB instance.
3. Create the Read Replica using one of the methods in the following sections.
4. Check the progress of the Read Replica creation using, for example, the **DescribeDBInstances** API operation. Once the Read Replica is available, unlock the tables of the source DB instance and resume normal database operations.

AWS Management Console

To create a Read Replica from a source MySQL or PostgreSQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **DB Instances**.

3. In the **My DB Instances** pane, right click the MySQL or PostgreSQL DB instance that you want to use as the source for a Read Replica and select **Create Read Replica**.
4. In the **DB Instance Identifier** text box, type a name for the Read Replica. Adjust other settings as needed.
5. In the **Destination Region** box, specify the region for the Read Replica if it is different than the region of the source DB instance.
6. In the **Destination DB Subnet Group** box, specify a DB subnet group associated with a VPC if you want the Read Replica to be created in that VPC. Leave the box empty if you want the Read Replica to be created outside of any VPC. The VPC and DB subnet group must exist in the destination region. Within a given region, all Read Replicas created from the same source DB instance must be either:

   - All created in the same VPC.
   - All created outside of any VPC.

7. Click **Yes, Create Read Replica**.

CLI

To create a Read Replica from a source MySQL or PostgreSQL DB instance

- Use the `rds-create-db-instance-read-replica` command.

API

To create a Read Replica from a source MySQL or PostgreSQL DB instance

- Call `CreateDBInstanceReadReplica`.

Promoting a Read Replica to Be a DB Instance

You can promote a MySQL or PostgreSQL Read Replica into a standalone, Single-AZ DB instance. When you promote a Read Replica, the DB instance will be rebooted before it becomes available.

There are several reasons you might want to convert a Read Replica into a Single-AZ DB instance:

- **Perform DDL operations (MySQL only)** : DDL operations, such as creating or re-building indexes, can take time and impose a significant performance penalty on your DB instance. You can perform these operations on a MySQL Read Replica once the Read Replica is in sync with its source DB instance. Then you can promote the Read Replica and direct your applications to use the promoted instance.
- **Sharding:** Sharding embodies the "share-nothing" architecture and essentially involves breaking a large database into several smaller databases. Common ways to split a database include 1)splitting tables that are not joined in the same query onto different hosts or 2)duplicating a table across multiple hosts and then using a hashing algorithm to determine which host receives a given update. You can create Read Replicas corresponding to each of your “shards” (smaller databases) and promote them when you decide to convert them into “standalone” shards. You can then carve out the key space (if you are splitting rows) or distribution of tables for each of the shards depending on your requirements.
- **Implement Failure Recovery** - You can use Read Replica promotion as a data recovery scheme if the source DB instance fails; however, if your use case requires synchronous replication, automatic failure detection, and failover, we recommend that you run your DB instance as a Multi-AZ deployment instead. If you are aware of the ramifications and limitations of asynchronous replication and you still

want to use Read Replica promotion for data recovery, you would first create a Read Replica and then monitor the source DB instance for failures. In the event of a failure, you would do the following:

1. Promote the Read Replica.
2. Direct database traffic to the promoted DB instance.
3. Create a replacement Read Replica with the promoted DB instance as its source.

You can perform all of these operations using the Amazon Relational Database Service API Reference, and you can automate the process by using the Amazon Simple Workflow Service Developer Guide.

The new DB instance that is created when you promote a Read Replica retains the backup retention period, backup window period, and parameter group of the former Read Replica source. The promotion process can take several minutes or longer to complete, depending on the size of the Read Replica. Once you promote the Read Replica into a Single-AZ DB instance, it is just like any other Single-AZ DB instance. For example, you can convert the new DB instance into a Multi-AZ DB instance, and you can create Read Replicas from it. You can also take DB snapshots and perform point-in-time restore operations. Because the promoted DB instance is no longer a Read Replica, you cannot use it as a replication target. If a source DB instance has several Read Replicas, promoting one of the Read Replicas to a DB instance has no effect on the other replicas.

We recommend that you enable automated backups on your Read Replica before promoting the Read Replica. This ensures that no backup is taken during the promotion process. Once the instance is promoted to a primary instance, backups are taken based on your backup settings.

The following steps show the general process for promoting a Read Replica to a Single-AZ DB instance.

1. Stop any transactions from being written to the Read Replica source DB instance, and then wait for all updates to be made to the Read Replica. Database updates occur on the Read Replica after they have occurred on the source DB instance, and this replication "lag" can vary significantly. Use the Replica Lag metric to determine when all updates have been made to the Read Replica.
2. (MySQL only) If you need to make changes to the MySQL Read Replica, you must the set the **read_only** parameter to **0** in the DB parameter group for the Read Replica. You can then perform all needed DDL operations, such as creating indexes, on the Read Replica. Actions taken on the Read Replica do not affect the performance of the source DB instance.
3. Promote the Read Replica by using the **Promote Read Replica** option on the Amazon RDS console, the CLI command `rds-promote-read-replica`, or the `PromoteReadReplica` API operation.

   Note
   The promotion process takes a few minutes to complete. When you promote a Read Replica, replication is stopped and the Read Replica is rebooted. When the reboot is complete, the Read Replica is available as a Single-AZ DB instance.

AWS Management Console

To promote a Read Replica to a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the Amazon RDS Console, click **Read Replicas**.
3. In the **Read Replicas** pane, select the check box beside the Read Replica that you want to promote.
4. Click **Promote Read Replica**.
5. In the **Promote Read Replica** dialog box, enter the backup retention period and the backup window for the new promoted DB instance.
6. When the settings are as you want them, click **Continue**.
7. On the acknowledgment page, click **Yes, Promote**.

CLI

To promote a Read Replica to a DB instance

- Use the `rds-promote-read-replica` command.

API

To promote a Read Replica to a DB instance

- Call `PromoteReadReplica`.

Replicating a Read Replica Across Regions (MySQL only)

You can create a MySQL Read Replica in a different region than the source DB instance to improve your disaster recovery capabilities, scale read operations into a region closer to end users, or make it easier to migrate from a data center in one region to a data center in another region. Creating a MySQL Read Replica in a different region than the source instance is very similar to creating a replica in the same region. You run the create Read Replica command in the region where you want the Read Replica, and specify the Amazon Resource Name (ARN) of the source DB instance.

Cross-Region Replication Considerations

All of the considerations for performing replication within a region apply to cross-region replication. The following extra considerations apply when replicating between regions:

- You can only replicate between regions when using Amazon RDS DB instances of MySQL 5.6.
- You can only cross one regional boundary in a given replication chain. You can create a cross-region Amazon RDS Read Replica from:
  - A source Amazon RDS DB instance that is not a Read Replica of another Amazon RDS DB instance.
  - An Amazon RDS DB instance that is a Read Replica of an on-premises or Amazon EC2 instance of MySQL that is not in Amazon RDS.
- You cannot set up a replication channel into or out of the AWS GovCloud (US) region.
- Cross-region replication is currently not supported for the EU (Frankfurt) region because you cannot copy a snapshot to or from that region. For more information, see Copying a DB Snapshot (p. 504).
- You should expect to see some higher level of lag time for any Read Replica that is in a different region than the source instance, due to the longer network channels between regional data centers.
- Within a region, all cross-region replicas created from the same source DB instance must either be in the same Amazon VPC or be outside of a VPC. For those Read Replicas, any of the create Read Replica commands that specify the `--db-subnet-group-name` parameter must specify a DB subnet group from the same VPC.
- You can create a cross-region Read Replica in a VPC from a source DB instance that is not in an VPC. You can also create a cross-region Read Replica that is not in an VPC from a source DB instance that is in a VPC.

Cross Region Replication Costs

The data transferred for cross region replication incurs Amazon RDS data transfer charges. These cross region replication actions generate charges for the data transferred out of the source region:

- When you create the Read Replica, Amazon RDS takes a snapshot of the source instance and transfers the snapshot to the Read Replica region.
- For each data modification made in the source databases, Amazon RDS transfers data from the source region to the Read Replica region.

For more information about Amazon RDS data transfer pricing, go to Amazon Relational Database Service Pricing.

You can reduce your data transfer costs by reducing the number of cross region Read Replicas you create. For example, if you have a source DB instance in one region and want to have three Read Replicas in another region, only create one of the Read Replicas from the source DB instance, and then create the other two replicas from the first Read Replica instead of the source. For example, if you have `source-instance-1` in one region, you can:

- Create `read-replica-1` in the new region, specifying `source-instance-1` as the source.
- Create `read-replica-2` from `read-replica-1`.
- Create `read-replica-3` from `read-replica-1`.

In this example, you will only be charged for the data transferred from `source-instance-1` to `read-replica-1`. You will not be charged for the data transferred from `read-replica-1` to the other two replicas because they are all in the same region. If you created all three replicas directly from `source-instance-1`, you would be charged for the data transfers to all three replicas.

Examples

Example Create Cross Region Read Replica Outside of any VPC

This is an example of creating a Read Replica in us-west-2 from a source DB instance in us-east-1. The Read Replica is created outside of a VPC:

```
PROMPT> rds-create-db-instance-read-replica SimCoProd01Replica01 --region us-
west-2 --source-db-instance-identifier arn:aws:rds:us-east-
1:123456789012:db:SimcoProd01
```

Example Create Cross Region Read Replica in a VPC

This is an example of creating a Read Replica in us-west-2 from a source DB instance in us-east-1. The Read Replica is created in the VPC associated with the specified DB subnet group:

```
PROMPT> rds-create-db-instance-read-replica SimCoProd01Replica01 --region us-
west-2 --db-subnet-group-name my-us-west-2-subnet --source-db-instance-identi
fier arn:aws:rds:us-east-1:123456789012:db:SimcoProd01
```

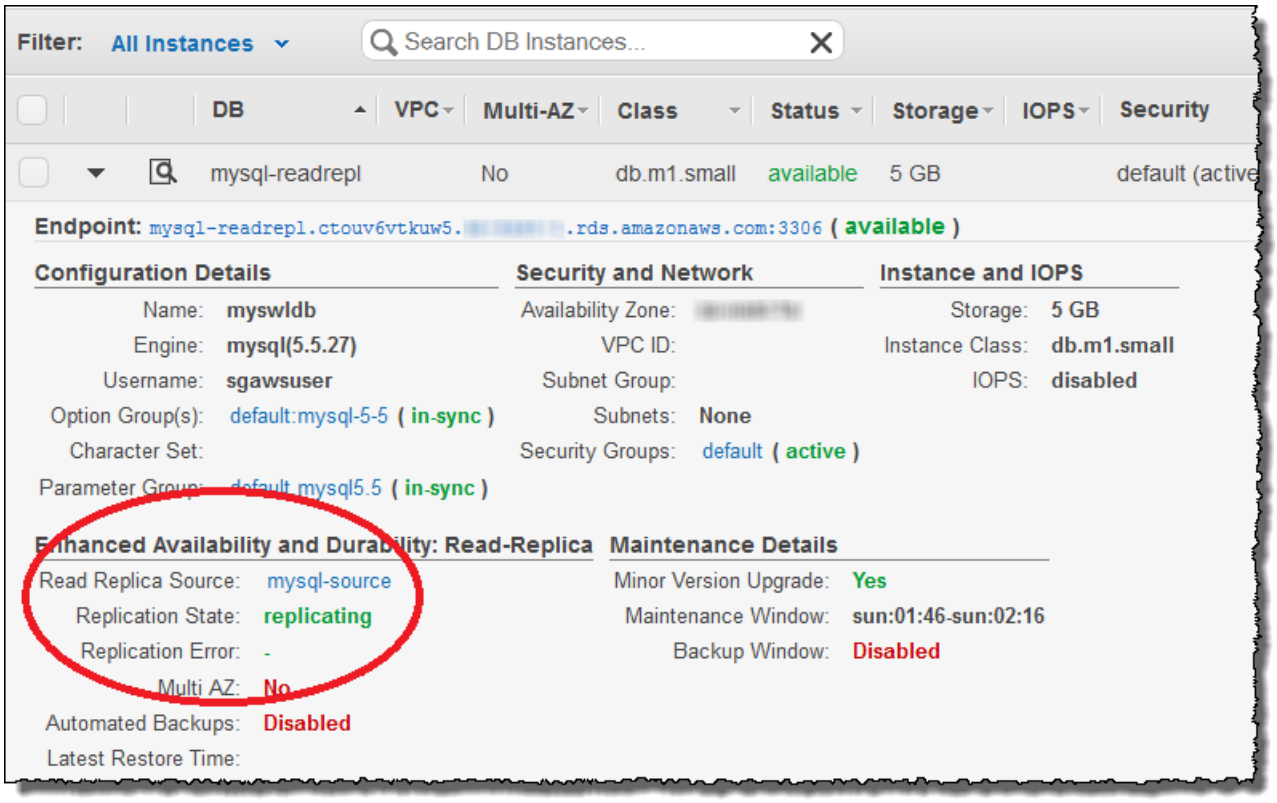
Cross Region Replication Process

Amazon RDS uses the following process to create a cross region Read Replica. Depending on the regions involved and the amount of data in the databases, this process could take hours to complete. You can use this information to determine how far the process has proceeded when you create a cross region Read Replica:

1. Amazon RDS begins configuring the source DB instance as a replication source and sets the status to *modifying*.
2. Amazon RDS begins setting up the specified Read Replica in the destination region and sets the status to *creating*.
3. Amazon RDS creates an automated DB snapshot of the source DB instance in the source region. The format of the DB snapshot name is rds:<InstanceID>-<timestamp>, where <InstanceID> is the identifier of the source instance, and <timestamp> is the date and time the copy started. For example, `rds:mysourceinstance-2013-11-14-09-24` was created from the instance `mysourceinstance` at `2013-11-14-09-24`. During this phase, the source DB instance status remains *modifying*, the Read Replica status remains *creating*, and the DB snapshot status is *creating*. The progress column of the DB snapshot page in the console reports how far the DB snapshot creation has progressed. When the DB snapshot is complete, the status of both the DB snapshot and source DB instance are set to *available*.
4. Amazon RDS begins a cross region snapshot copy for the initial data transfer. The snapshot copy is listed as an automated snapshot in the destination region with a status of *creating*. It has the same name as the source DB snapshot. The progress column of the DB snapshot display indicates how far the copy has progressed. When the copy is complete, the status of the DB snapshot copy is set to *available*.
5. Amazon RDS then uses the copied DB snapshot for the initial data load on the Read Replica. During this phase, the Read Replica will be in the list of DB instances in the destination, with a status of *creating*. When the load is complete, the Read Replica status is set to *available*, and the DB snapshot copy is deleted.
6. When the Read Replica reaches the available status, Amazon RDS starts by replicating the changes made to the source instance since the start of the create Read Replica operation. During this phase, the replication lag time for the Read Replica will be greater than 0. You can monitor this in Amazon CloudWatch by viewing the Amazon RDS `ReplicaLag` metric. The ReplicaLag metric reports the value of the `Seconds_Behind_Master` field of the MySQL `SHOW SLAVE STATUS` command. For more information, see SHOW SLAVE STATUS. When the `ReplicaLag` metric reaches 0, the replica has caught up to the source DB instance. If the `ReplicaLag` metric returns -1, then replication is currently not active. `ReplicaLag` = -1 is equivalent to `Seconds_Behind_Master` = `NULL`. Common causes for `ReplicaLag` returning -1 are the following:
   - A network outage.
   - Writing to tables with indexes on a Read Replica. If the `read_only` parameter is not set to 0 on the Read Replica, it can break replication.
   - Using a non-transactional storage engine such as MyISAM. Replication is only supported for the InnoDB storage engine.

Monitoring Read Replication

You can monitor the status of a Read Replica in several ways. The Amazon RDS console shows the status of a Read Replica; you can also see the status of a Read Replica using the CLI command `rds-describe-db-instances` or the API action `DescribeDBInstances`.



The status of a Read Replica can be one of the following:

- **Replicating—**The Read Replica is replicating successfully.
- **Error—**An error has occurred with the replication. Check the **Replication Error** field in the Amazon RDS console or the event log to determine the exact error. For more information about troubleshooting a replication error, see Troubleshooting a MySQL Read Replica Problem (p. 483).
- **Stopped—**(MySQL only) Replication has stopped because of a customer initiated request.
- **Terminated—**The Read Replica has lagged the source DB instance for more than the backup retention period due to replication errors and is terminated. The Read Replica is still accessible for read operations but cannot synchronize with the source instance.

If replication errors occur in a Read Replica for more than the backup retention period, replication is terminated to prevent increased storage requirements and long failover times. Broken replication can effect storage because the logs can grow in size and number due to the high volume of errors messages being written to the log. Broken replication can also affect failure recovery due to the time Amazon RDS requires to maintain and process the large number of logs during recovery.

You can monitor how far a MySQL Read Replica is lagging the source DB instance by viewing the **Seconds_Behind_Master** data returned by the MySQL **Show Slave Status** command, or the CloudWatch **Replica Lag** statistic. If a replica lags too far behind for your environment, consider deleting and recreating the Read Replica. Also consider increasing the scale of the Read Replica to speed replication.

You can monitor PostgreSQL Read Replica lag by viewing the CloudWatch **Replica Lag** statistic or by running the following command from the PostgreSQL source DB instance:

```
select now() - pg_last_xact_replay_timestamp() AS replication_delay;
```

Troubleshooting a MySQL Read Replica Problem

MySQL's replication technology is asynchronous. Because it is asynchronous, occasional `BinLogDiskUsage` increases on the source DB instance and `ReplicaLag` on the Read Replica are to be expected. For example, a high volume of writes to the source DB instance can occur in parallel, while writes to the Read Replica are serialized using a single I/O thread, can lead to a lag between the source instance and Read Replica. For more information about read-only replicas in the MySQL documentation, see Replication Implementation Details.

There are several things you can do to reduce the lag between updates to a source DB instance and the subsequent updates to the Read Replica, such as:

- Sizing a Read Replica to have a storage size and DB instance class comparable to the source DB instance.
- Ensuring that parameter settings in the DB parameter groups used by the source DB instance and the Read Replica are compatible. For more information and an example, see the discussion of the `max_allowed_packet` parameter later in this section.

Amazon RDS monitors the replication status of your Read Replicas and updates the `Replication State` field of the Read Replica instance to `Error` if replication stops for any reason, such as DML queries being run on your Read Replica that conflict with the updates made on the source DB instance. You can review the details of the associated error thrown by the MySQL engine by viewing the `Replication Error` field. Events that indicate the status of the Read Replica are also generated, including RDS-EVENT-0045 (p. 579), RDS-EVENT-0046 (p. 579), and RDS-EVENT-0047 (p. 578). For more information about events and subscribing to events, see Using Amazon RDS Event Notification (p. 575). If the MySQL error message is returned, review the error number in the MySQL error message documentation.

One common issue that can cause replication errors is when the value for the `max_allowed_packet` parameter for a Read Replica is less than the `max_allowed_packet` parameter for the source DB instance. The `max_allowed_packet` parameter is a custom parameter that you can set in a DB parameter group that is used to specify the maximum size of DML that can be executed on the database. If the `max_allowed_packet` parameter value in the DB parameter group associated with a source DB instance is smaller than the `max_allowed_packet` parameter value in the DB parameter group associated with the source's Read Replica, the replication process can throw an error (Packet bigger than 'max_allowed_packet' bytes) and stop replication. You can fix the error by having the source and Read Replica use DB parameter groups with the same `max_allowed_packet` parameter values.

Other common situations that can cause replication errors include:

- Writing to tables on a Read Replica. If you are creating indexes on a Read Replica, you need to have the `read_only` parameter set to *0* to create the indexes. If you are writing to tables on the Read Replica, it may break replication.
- Using a non-transactional storage engine such as MyISAM. Read replicas require a transactional storage engine. Replication is only supported for the InnoDB storage engine.
- Using unsafe non-deterministic queries such as `SYSDATE()`. For more information, see Determination of Safe and Unsafe Statements in Binary Logging.

If you decide that you can safely skip an error, you can follow the steps described in the section Skipping the Current Replication Error (p. 168). Otherwise, you can delete the Read Replica and create a instance

using the same DB instance identifier so that the endpoint remains the same as that of your old Read Replica. If a replication error is fixed, the `Replication State` changes to *replicating*.

Troubleshooting a PostgreSQL Read Replica Problem

The PostgreSQL parameter, `wal_keep_segments`, dictates how many Write Ahead Log (WAL) files are kept to provide data to the Read Replicas. The parameter value species the number of logs to keep. If you set the parameter value too low, you can cause a Read Replica to fall so far behind that streaming replication stops. In this case, Amazon RDS will report a replication error and begin recovery on the Read Replica by replaying the source DB instance's archived WAL logs. This recovery process continues until the Read Replica has caught up enough to continue streaming replication.

The PostgreSQL log will show when Amazon RDS is recovering a Read Replica that is this state by replaying archived WAL files.

```
  2014-11-07 19:01:10 UTC::@:[23180]:DEBUG:  switched WAL source from archive
to stream after
      failure 2014-11-07 19:01:10 UTC::@:[11575]:LOG:  started streaming WAL
from primary at
     1A/D3000000 on timeline 1 2014-11-07 19:01:10 UTC::@:[11575]:FATAL:  could
 not receive
      data from WAL stream: ERROR:  requested WAL segment
000000010000001A000000D3 has already been
     removed 2014-11-07 19:01:10 UTC::@:[23180]:DEBUG:  could not restore file

      "00000002.history" from archive: return code 0 2014-11-07 19:01:15
      UTC::@:[23180]:DEBUG:  switched WAL source from stream to archive after
failure
      recovering 000000010000001A000000D3 2014-11-07 19:01:16
UTC::@:[23180]:LOG:  restored log file "000000010000001A000000D3"
        from archive
```

Once Amazon RDS has replayed enough archived WAL files on the replica to catch up and allow the Read Replica to begin streaming again, PostgreSQL will resume streaming and write a similar line to the following to the log file:

```
  2014-11-07 19:41:36 UTC::@:[24714]:LOG:  started streaming WAL from primary
at 1B/B6000000
      on timeline 1
```

You can determine how many WAL files you should keep by looking at the checkpoint information in the log. The PostgreSQL log shows the following information at each checkpoint. By looking at the "# recycled" transaction log files of these log statements, a user can understand how many transaction files will be recycled during a time range and use this information to tune the `wal_keep_segments` parameter.

```
  2014-11-07 19:59:35 UTC::@:[26820]:LOG:  checkpoint complete: wrote 376 buffers
 (0.2%); 0
      transaction log file(s) added, 0 removed, 1 recycled; write=35.681 s,
sync=0.013 s,
```

```
    total=35.703 s; sync files=10, longest=0.013 s, average=0.001 s
```

For example, if the PostgreSQL log shows that 35 files are recycled from the "checkpoint completed" log statements within a 5 minute time frame, we know that with this usage pattern a Read Replica relies on 35 transaction files in 5 minutes and could not survive 5 minutes in a non-streaming state if the source DB instance is set to the default `wal_keep_segments` parameter value of 32.

Tagging Amazon RDS Resources

What You Should Know About Amazon RDS Resource Tags

You can use Amazon RDS tags to add metadata to your Amazon RDS resources. In addition, these tags can be used with IAM policies to manage access to Amazon RDS resources and to control what actions can be applied to the Amazon RDS resources. Finally, these tags can be used to track costs by grouping expenses for similarly tagged resources.

All Amazon RDS resources can be tagged:

- DB instances
- Read replicas
- DB snapshots
- Reserved DB instances
- Event subscriptions
- DB option groups
- DB parameter groups
- DB security groups
- DB subnet groups

For information on managing access to tagged resources with IAM policies, see Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources (p. 88).

An Amazon RDS tag is a name-value pair that you define and associate with an Amazon RDS resource. The name is referred to as the key. Supplying a value for the key is optional. You can use tags to assign arbitrary information to an Amazon RDS resource. A tag key could be used, for example, to define a category, and the tag value could be a item in that category. For example, you could define a tag key of “project” and a tag value of “Salix,” indicating that the Amazon RDS resource is assigned to the Salix project. You could also use tags to designate Amazon RDS resources as being used for test or production by using a key such as environment=test or environment =production. We recommend that you use a consistent set of tag keys to make it easier to track metadata associated with Amazon RDS resources.

Use tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. Then, to see the cost of combined resources, organize your billing information according to resources with the same tag key values. For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services. For more information, see Cost Allocation and Tagging in *About AWS Billing and Cost Management*.

Each Amazon RDS resource has a tag set, which contains all the tags that are assigned to that Amazon RDS resource. A tag set can contain as many as ten tags, or it can be empty. If you add a tag to an Amazon RDS resource that has the same key as an existing tag on resource, the new value overwrites the old value.

AWS does not apply any semantic meaning to your tags; tags are interpreted strictly as character strings. AWS does not automatically set any tags on Amazon RDS resources.

The following list describes the characteristics of a DB instance tag.

- The tag key is the required name of the tag. The string value can be from 1 to 128 Unicode characters in length and cannot be prefixed with "aws:" or "rds:". The string may contain only the set of Unicode letters, digits, white-space, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$").

- The tag value is an optional string value of the tag. The string value can be from 1 to 256 Unicode characters in length and cannot be prefixed with "aws:" or "rds:". The string may contain only the set of Unicode letters, digits, white-space, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$").

  Values do not have to be unique in a tag set and can be null. For example, you can have a key-value pair in a tag set of project/Trinity and cost-center/Trinity.

You can use the AWS Management Console, the command line interface, or the Amazon RDS API to add, list, and delete tags on Amazon RDS resources. When using the command line interface or the Amazon RDS API, you must provide the Amazon Resource Name (ARN) for the Amazon RDS resource you want to work with. For more information about constructing an ARN, see Constructing an Amazon RDS Amazon Resource Name (ARN) (p. 493).

Note that tags are cached for authorization purposes. Because of this, additions and updates to tags on Amazon RDS resources may take several minutes before they are available.

Copying Tags

When you create or restore a DB instance, you can specify that the tags from the DB instance are copied to snapshots of the DB instance. Copying tags ensures that the metadata for the DB snapshots matches that of the source DB instance and any access policies for the DB snapshot also match those of the source DB instance. Tags are not copied by default.

You can specify that tags are copied to DB snapshots for the following actions:

- Creating a DB instance.
- Restoring a DB instance.
- Creating a Read Replica.
- Copying a DB snapshot.

Note
If you include a value for the `--tag-key` parameter of the rds-create-db-snapshot CLI command (or supply at least one tag to the CreateDBSnapshot API action) then RDS will not copy tags from the source DB instance to the new DB snapshot. This functionality applies even if the source DB instance has the `--copy-tags-to-snapshot` (**CopyTagsToSnapshot**) option enabled. If you take this approach, you can create a copy of a DB instance from a DB snapshot without adding tags that don't apply to the new DB instance. Once you have created your DB snapshot using the **rds-create-db-snapshot** CLI command (or the **CreateDBSnapshot** API action) you can then add tags as described later in this topic.

AWS Management Console

The process to tag an Amazon RDS resource is similar for all resources. The following example shows how to tag an Amazon RDS DB instance.

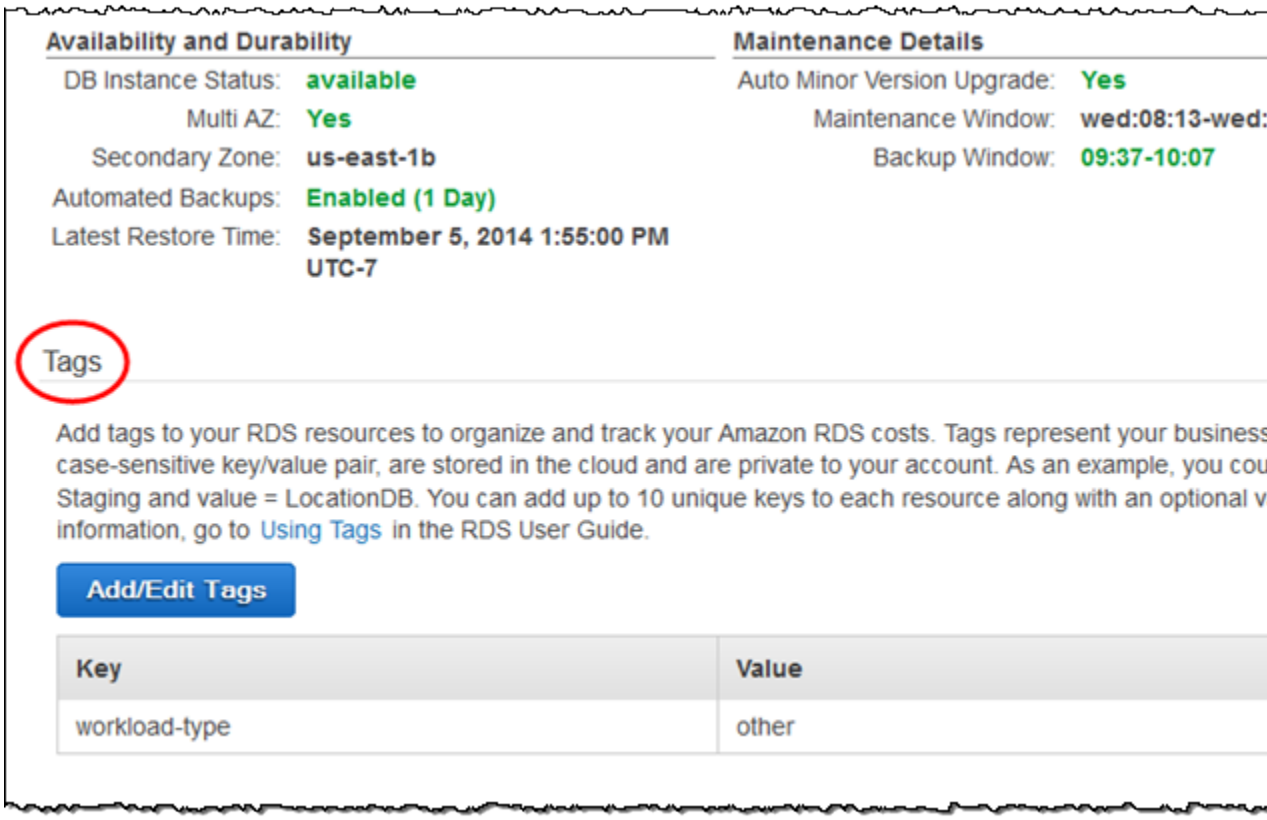
To add a tag to a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Instances**.

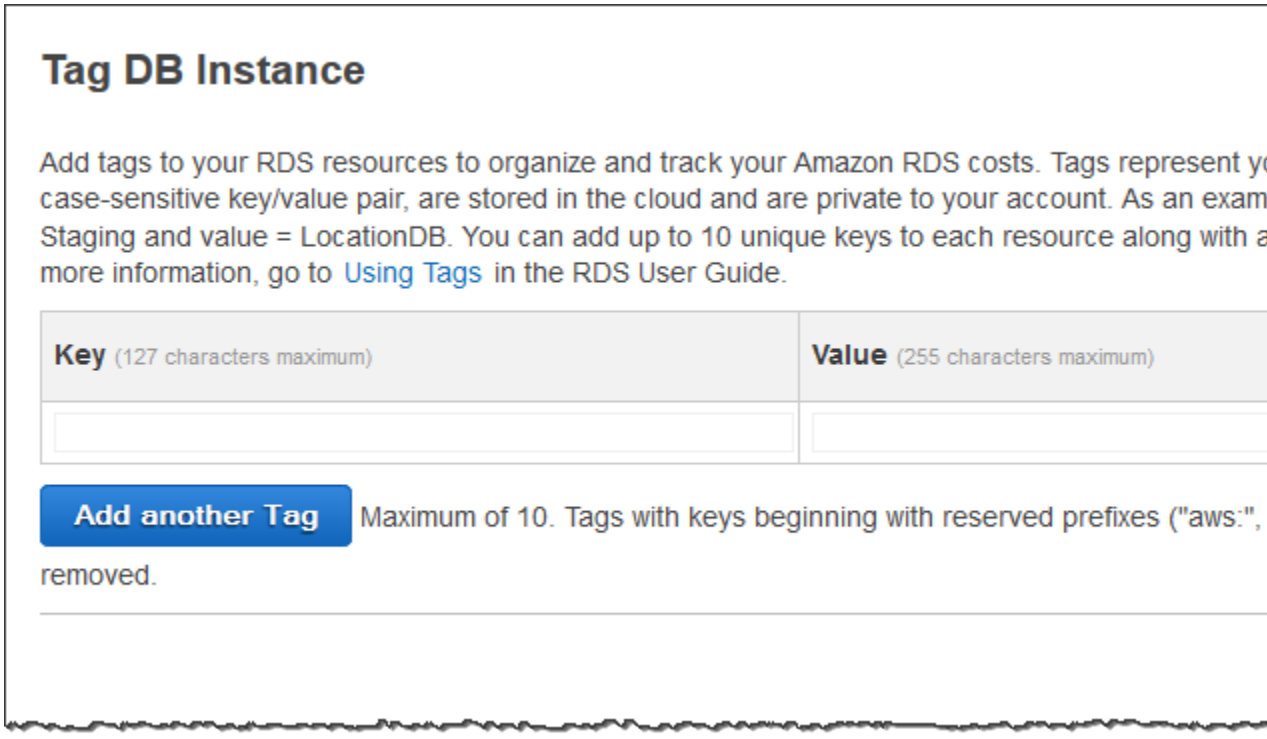
Note
To filter the list of DB instances in the **DB Instances** pane, in the box beside the **Viewing** box, type a text string. Only DB instances that contain the string will appear.

3. Select the check box for the DB instance that you want to tag.
4. Click the details icon.

5. In the details pane, scroll down to **Tags**.


Availability and Durability
DB Instance Status: available
Multi AZ: Yes
Secondary Zone: us-east-1b
Automated Backups: Enabled (1 Day)
Latest Restore Time: September 5, 2014 1:55:00 PM UTC-7
Maintenance Details
Auto Minor Version Upgrade: Yes
Maintenance Window: wed:08:13-wed:
Backup Window: 09:37-10:07
Tags
Add tags to your RDS resources to organize and track your Amazon RDS costs. Tags represent your business
case-sensitive key/value pair, are stored in the cloud and are private to your account. As an example, you cou
Staging and value = LocationDB. You can add up to 10 unique keys to each resource along with an optional v
information, go to Using Tags in the RDS User Guide.
Add/Edit Tags
Key
Value
workload-type
other

6. Click **Add/Edit Tags**.


Tag DB Instance
Add tags to your RDS resources to organize and track your Amazon RDS costs. Tags represent yo
case-sensitive key/value pair, are stored in the cloud and are private to your account. As an exam
Staging and value = LocationDB. You can add up to 10 unique keys to each resource along with a
more information, go to Using Tags in the RDS User Guide.
Key (127 characters maximum)
Value (255 characters maximum)
Add another Tag
Maximum of 10. Tags with keys beginning with reserved prefixes ("aws:",
removed.

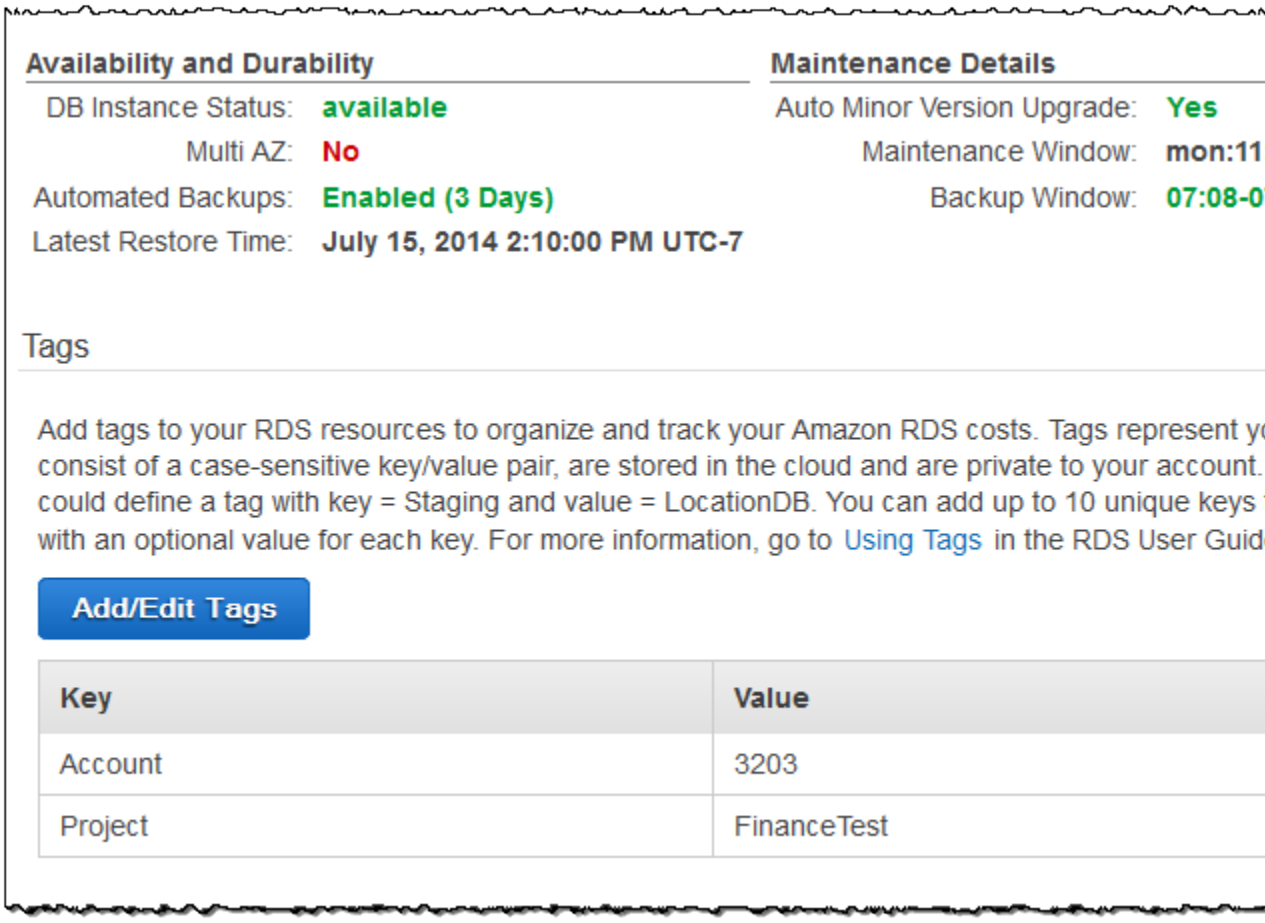
7. Type a name and value for the tag. Click **Save Tags**.

To delete a tag from a DB instance

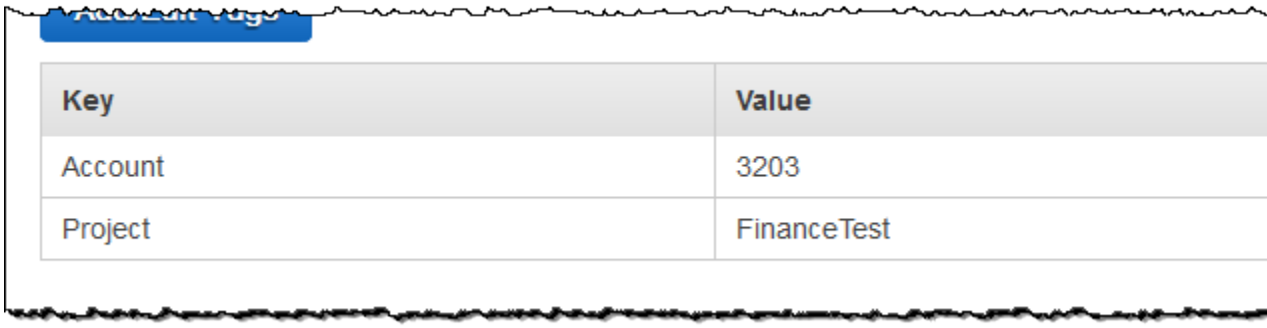
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Instances**.

   Note
   To filter the list of DB instances in the **DB Instances** pane, in the box beside the **Viewing** box, type a text string. Only DB instances that contain the string will appear.
3. Select the check box for the DB instance from which you want to remove a tag.
4. Click the details icon.

5. In the details pane, scroll down to **Tags**.


Availability and Durability
DB Instance Status: available
Multi AZ: No
Automated Backups: Enabled (3 Days)
Latest Restore Time: July 15, 2014 2:10:00 PM UTC-7
Maintenance Details
Auto Minor Version Upgrade: Yes
Maintenance Window: mon:11
Backup Window: 07:08-0
Tags
Add tags to your RDS resources to organize and track your Amazon RDS costs. Tags represent y
consist of a case-sensitive key/value pair, are stored in the cloud and are private to your account.
could define a tag with key = Staging and value = LocationDB. You can add up to 10 unique keys
with an optional value for each key. For more information, go to Using Tags in the RDS User Guid
Add/Edit Tags
Key
Value
Account
3203
Project
FinanceTest

6. Click the red "X" in the **Remove** column next to the tag you want to delete.



7. Click the **Save Tags** button.

CLI

To add, list, or remove tags for a DB instance

- To add a tag to an Amazon RDS resource, use the **rds-add-tag-to-resource** command.
- To list tags that are assigned to an Amazon RDS resource, use the **rds-list-tags-for-resource** command.

- To remove tags from an Amazon RDS resource, use the **rds-remove-tags-from-resource** command.

To learn more about how to construct the required ARN, see Constructing an Amazon RDS Amazon Resource Name (ARN) (p. 493)

API

To add, list, or remove tags for a DB instance

- To add a tag to an Amazon RDS resource, use the `AddTagsToResource` operation.
- To list tags that are assigned to an Amazon RDS resource, use the `ListTagsForResource`.
- To remove tags from an Amazon RDS resource, use the `RemoveTagsFromResource` operation.

To learn more about how to construct the required ARN, see Constructing an Amazon RDS Amazon Resource Name (ARN) (p. 493)

When working with XML using the Amazon RDS API, tags use the following schema:

```
<Tagging>
  <TagSet>
   <Tag>
    <Key>Project</Key>
    <Value>Trinity</Value>
   </Tag>
   <Tag>
    <Key>User</Key>
    <Value>Jones</Value>
   </Tag>
  </TagSet>
</Tagging>
```

The following table provides a list of the allowed XML tags and their characteristics. Note that values for Key and Value are case dependent. For example, project=Trinity and PROJECT=Trinity are two distinct tags.

Tagging element	Description
TagSet	A tag set is a container for all tags assigned to an Amazon RDS resource. There can be only one tag set per resource. You work with a TagSet only through the Amazon RDS API.
Tag	A tag is a user-defined key-value pair. There can be from 1 to 10 tags in a tag set.
Key	A key is the required name of the tag. The string value can be from 1 to 128 Unicode characters in length and cannot be prefixed with "rds:" or "aws:". The string may only contain only the set of Unicode letters, digits, white-space, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). Keys must be unique to a tag set. For example, you cannot have a key-pair in a tag set with the key the same but with different values, such as project/Trinity and project/Xanadu.

Tagging element	Description
Value	A value is the optional value of the tag. The string value can be from 1 to 256 Unicode characters in length and cannot be prefixed with "rds:" or "aws:". The string may only contain only the set of Unicode letters, digits, white-space, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). Values do not have to be unique in a tag set and can be null. For example, you can have a key-value pair in a tag set of project/Trinity and cost-center/Trinity.

Constructing an Amazon RDS Amazon Resource Name (ARN)

Resources that are created in Amazon Web Services are identified by a unique identifier call an Amazon Resource Name (ARN). If you use the CLI or Amazon RDS API to add, modify, or delete tags, you must supply the ARN of the resource you want to work with.

An ARN for an Amazon RDS resource uses the following syntax:

```
arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

- *<region>* is the AWS region ID where the Amazon RDS resource was created, such as us-west-2.

  The following table shows AWS region names and the value you should use when constructing an ARN.

Region	Name	Endpoint
US East (N. Virginia) region	us-east-1	https://rds.us-east-1.amazonaws.com
US West (N. California) region	us-west-1	https://rds.us-west-1.amazonaws.com
US West (Oregon) region	us-west-2	https://rds.us-west-2.amazonaws.com
EU (Ireland) region	eu-west-1	https://rds.eu-west-1.amazonaws.com
EU (Frankfurt) Region	eu-central-1	https://rds.eu-central-1.amazonaws.com
Asia Pacific (Tokyo) Region	ap-northeast-1	https://rds.ap-northeast-1.amazonaws.com
Asia Pacific (Singapore) Region	ap-southeast-1	https://rds.ap-southeast-1.amazonaws.com
Asia Pacific (Sydney) Region	ap-southeast-2	https://rds.ap-southeast-2.amazonaws.com
South America (Sao Paulo) Region	sa-east-1	https://rds.sa-east-1.amazonaws.com

Region	Name	Endpoint
China (Beijing) Region	cn-north-1	https://rds.cn-north-1.amazonaws.com
AWS GovCloud (US) Region	us-gov-west-1	https://rds.us-gov-west-1.amazonaws.com

- <*account number*> is your account number with dashes omitted. To find your account number, log into your AWS account at http://www.amazonaws.cn, click **My Account/Console**, and then click **My Account**.
- <*resourcetype*> is the type of Amazon RDS resource.

  The following table shows the resource type you should use when constructing an ARN for a particular Amazon RDS resource.

Resource Type	ARN Format
DB instance	arn:aws:rds:<*region*>:<*account*>:db:<*dbinstance name*>
Event subscription	arn:aws:rds:<*region*>:<*account*>:es:<*subscription name*>
DB option group	arn:aws:rds:<*region*>:<*account*>:og:<*option group name*>
DB parameter group	arn:aws:rds:<*region*>:<*account*>:pg:<*parameter group name*>
Reserved DB instance	arn:aws:rds:<*region*>:<*account*>:ri:<*reserve instance name*>
DB security group	arn:aws:rds:<*region*>:<*account*>:secgrp:<*security group name*>
DB snapshot	arn:aws:rds:<*region*>:<*account*>:snapshot:<*snapshot name*>
DB subnet group	arn:aws:rds:<*region*>:<*account*>:subgrp:<*subnet group name*>

- <*name*> is the resource identifier for the Amazon RDS resource.

The following table shows examples of ARNs for RDS resources with an AWS account of 123456789012, that were created in the US East (N. Virginia) region, and that have a resource name that begins with "my-":

Resource Type	Sample ARN
DB instance	arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
Event subscription	arn:aws:rds:us-east-1:123456789012:es:my-subscription
DB option group	arn:aws:rds:us-east-1:123456789012:og:my-option-group-oracle-tde
DB parameter group	arn:aws:rds:us-east-1:123456789012:pg:my-param-enable-logs
Reserved DB instance	arn:aws:rds:us-east-1:123456789012:ri:my-reserved-multiaz
DB security group	arn:aws:rds:us-east-1:123456789012:secgrp:my-public
DB snapshot	arn:aws:rds:us-east-1:123456789012:snapshot:my-mysql-snap-20130507
DB subnet group	arn:aws:rds:us-east-1:123456789012:subgrp:my-subnet-10

Related Topics

- Using AWS Identity and Access Management (IAM) to Manage Access to Amazon RDS Resources (p. 88)

Backing Up and Restoring

This section shows how to back up and restore a DB instance.

Topics

- Working With Automated Backups (p. 496)
- Creating a DB Snapshot (p. 499)
- Restoring From a DB Snapshot (p. 501)
- Copying a DB Snapshot (p. 504)
- Restoring a DB Instance to a Specified Time (p. 508)

Working With Automated Backups

Amazon RDS can automatically back up all of your DB instances. You can set the backup retention period when you create a DB instance. If you don't set the backup retention period, Amazon RDS uses a default period retention period of one day. You can modify the backup retention period; valid values are 0 (for no backup retention) to a maximum of 35 days. Manual snapshot limits (50 per region) do not apply to automated backups.

> **Important**
> An outage will occur if you change the backup retention period from 0 to a non-zero value or from a non-zero value to 0.

All automated backups are deleted and cannot be recovered when you delete a DB instance. Manual snapshots are not deleted. For information on pricing for storing manual snapshots long-term, see Amazon RDS Pricing.

In this example, you will enable and then disable backups for an existing DB instance called *mydbinstance*.

Disabling Automated Backups

You may want to temporarily disable automated backups in certain situations; for example, while loading large amounts of data.

> **Important**
> We highly discourage disabling automated backups because it disables point-in-time recovery. If you disable and then re-enable automated backups, you are only able to restore starting from the time you re-enabled automated backups.

In these examples, you disable automated backups for a DB instance by setting the backup retention parameter to 0.

AWS Management Console

To disable automated backups immediately

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **DB Instances**, and then select the check box next to the DB instance you want to modify.
3. Click the **Modify** button.

   The **Modify DB Instance** window appears.
4. Select **0** in the **Backup Retention Period** drop-down list box.
5. Check the **Apply Immediately** check box.
6. Click the **OK** button.

CLI

To disable automated backups immediately

1. Set the backup retention period to 0.

```
PROMPT>rds-modify-db-instance mydbinstance --backup-retention-period 0 --
apply-immediately
```

2. Call `rds-describe-db-instances` for the DB instance until the value for backup retention period is 0 and mydbinstance status is available.

```
PROMPT>rds-describe-db-instances mydbinstance --headers
```

API

To disable automated backups immediately

- Call `ModifyDBInstance` with the following parameters:
  - *`DBInstanceIdentifier`* = `mydbinstance`
  - *`BackupRetentionPeriod`* = `0`

Example

```
https://rds.amazonaws.com/
 ?Action=ModifyDBInstance
 &DBInstanceIdentifier=mydbinstance
 &BackupRetentionPeriod=0
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-14T17%3A48%3A21.746Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Enabling Automated Backups

If your DB instance doesn't have automated backups enabled, you can enable them at any time. The same request used to disable automated backups can be used to enable them by using a non-zero value for the backup retention period. When automated backups are enabled, a backup is immediately created.

All automated backups are deleted and cannot be recovered when you delete a DB instance. Manual snapshots are not deleted.

In this example, you enable automated backups for a DB instance by setting the backup retention period parameter for the DB instance to a non-zero value (in this case, 3).

AWS Management Console

To enable automated backups immediately

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **DB Instances**, and then select the check box next to the DB instance you want to modify.
3. Click the **Modify** button or right-click the DB instance and select **Modify** from the context menu.

   The **Modify DB Instance** window appears.

4. Select **3** in the **Backup Retention Period** drop-down list box.
5. Check the **Apply Immediately** check box.
6. Click the **OK** button.

CLI

To enable automated backups immediately

In this example, we will enable automated backups by setting the backup retention period to 3.

- Set the backup retention period to 3.

```
PROMPT>rds-modify-db-instance mydbinstance --backup-retention-period 3 --
apply-immediately
```

API

To enable automated backups immediately

- Call `ModifyDBInstance` with the following parameters:
  - *`DBInstanceIdentifier`* = `mydbinstance`
  - *`BackupRetentionPeriod`* = `3`
  - *`ApplyImmediately`* = `true`

Example

```
https://rds.amazonaws.com/
 ?Action=ModifyDBInstance
 &DBInstanceIdentifier=mydbinstance
 &BackupRetentionPeriod=3
 &ApplyImmediately=true
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-14T17%3A48%3A21.746Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Related Topics

- Restoring a DB Instance to a Specified Time (p. 508)
- DB Instance Backups (p. 73)

Creating a DB Snapshot

When you create a DB snapshot, you need to identify which DB instance you are going to back up, and then give your DB snapshot a name so you can restore from it later.

> **Note**
> Creating a DB snapshot creates a backup of your DB instance. Creating this backup on a Single-AZ DB instance results in a brief I/O suspension that typically lasting no more than a few minutes. Multi-AZ DB instances are not affected by this I/O suspension since the backup is taken on the standby.

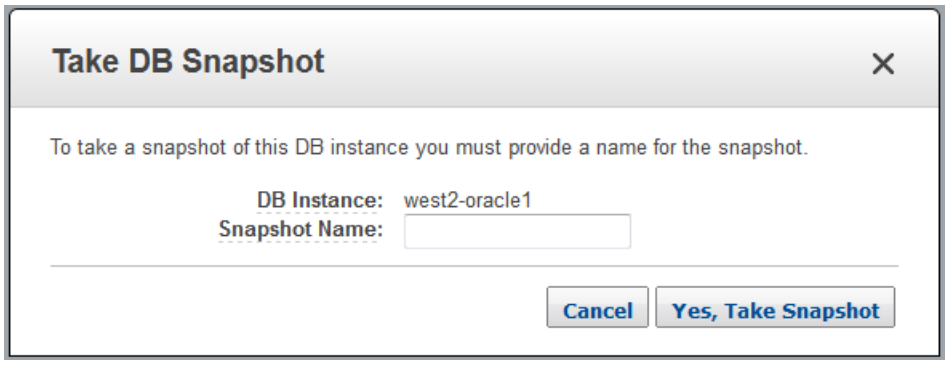
In this example, you create a DB snapshot called *mydbsnapshot* for a DB instance called *mydbinstance*.

AWS Management Console

To create a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **DB Instances**.
3. Click **Instance Actions**, and then click **Take DB Snapshot**.

   The **Take DB Snapshot** window appears.
4. Type the name of the snapshot in the **Snapshot Name** text box.


Take DB Snapshot
To take a snapshot of this DB instance you must provide a name for the snapshot.
DB Instance:
west2-oracle1
Snapshot Name:
Cancel
Yes, Take Snapshot

5. Click **Yes, Take Snapshot**.

CLI

To create a DB snapshot

- Use the command `rds-create-db-snapshot` to create a database snapshot.

```
PROMPT>rds-create-db-snapshot -i mydbinstance -s mydbsnapshot
```

The output from this command should look similar to the following:

```
DBSNAPSHOT  mydbsnapshot  mydbinstance  2009-10-21T01:54:49.521Z  MySQL
  50
```

```
creating  sa  5.1.57 general-public-license
```

API

To create a DB snapshot

- Call `CreateDBSnapshot` with the following parameters:
  - *`DBSnapshotIdentifier`* = `mydbsnapshot`
  - *`DBInstanceIdentifier`* = `mydbinstance`

Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=CreateDBSnapshot
   &DBInstanceIdentifier=mydbinstance
   &DBSnapshotIdentifier=mydbsnapshot
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140423/us-east-1/rds/aws4_request
   &X-Amz-Date=20140423T161105Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signature=e9649af6edcf
bab4016f04d72e1b7fc16d8734c37477afcf25b3def625484ed2
```

Related Topics

- Restoring From a DB Snapshot (p. 501)
- Copying a DB Snapshot (p. 504)
- DB Instance Backups (p. 73)

Restoring From a DB Snapshot

You must create a DB snapshot before you can restore a DB instance from one. When you restore the DB instance, you provide the name of the DB snapshot to restore from, and then provide a name for the new DB instance that is created from the restore. You cannot restore from a DB snapshot to an existing DB instance; a new DB instance is created when you restore.

When you restore a DB instance, only the default DB parameter and security groups are associated with the restored instance. As soon as the restore is complete, you should associate the custom DB parameter or security group you used by the instance you restored from. You must apply these changes explicitly using the RDS console's *Modify* command, the `ModifyDBInstance` API, or the `rds-modify-db-instance` command line tool, once the DB instance is available. We recommend that you retain parameter groups for any DB snapshots you have so that you can associate a restored instance with the correct parameter file.

> **Note**
> If you use Oracle GoldenGate, always retain the parameter group with the `compatible` parameter. If you restore an instance from a DB snapshot, you must modify the restored instance to use the parameter group that has a matching or greater `compatible` parameter value. This should be done as soon as possible after the restore action, and will require a reboot of the instance.

The option group associated with the DB snapshot is associated with the restored DB instance once it is created. For example, if the DB snapshot you are restoring from uses Oracle Transparent Data Encryption, the restored DB instance will use the same option group, which had the TDE option. When an option group is assigned to a DB instance, it is also linked to the supported platform the DB instance is on, either VPC or EC2-Classic (non-VPC). Furthermore, if a DB instance is in a VPC, the option group associated with the instance is linked to that VPC. This means that you cannot use the option group assigned to a DB instance if you attempt to restore the instance into a different VPC or onto a different platform. If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance. Note that with persistent or permanent options, such as Oracle TDE, you must create a new option group that includes the persistent or permanent option when restoring a DB instance into a different VPC.

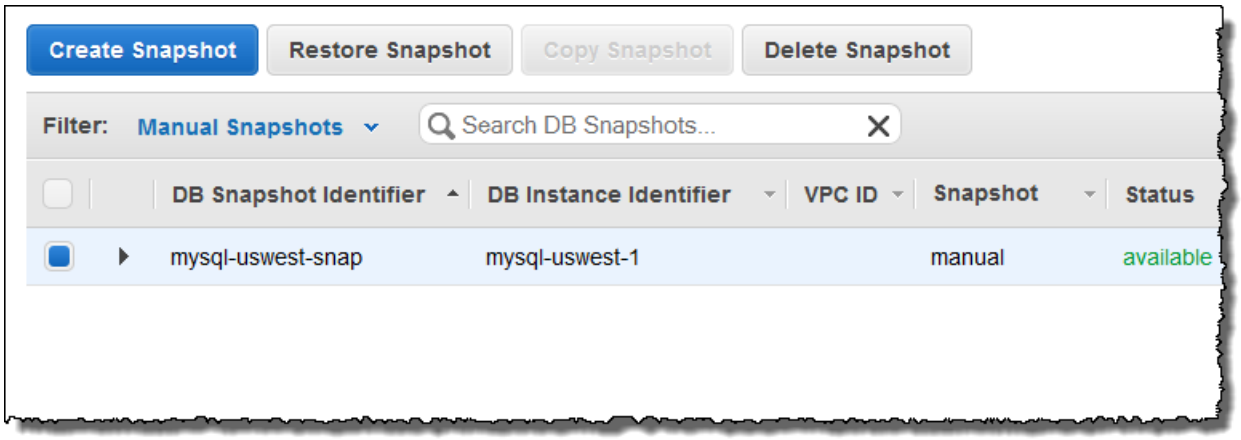
You can change to a different edition of the DB engine when restoring from a DB snapshot only if the DB snapshot has the required storage allocated for the new edition. For example, to change from SQL Server Web Edition to SQL Server Standard Edition, the DB snapshot must have been created from a SQL Server DB instance that had at least 200 GB of allocated storage, which is the minimum allocated storage for SQL Server Standard edition.

In this example, you restore from a previously created DB snapshot called *mydbsnapshot* and create a new DB instance called *mynewdbinstance*.

AWS Management Console

To restore a DB instance from a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Snapshots**.
3. Click on the DB snapshot that you want to restore from.



4. Click **Restore Snapshot**.

   The **Restore DB Instance** window appears.
5. Type the name of the restored DB instance in the **DB Instance Identifier** text box.
6. Click the **Launch DB Instance** button.

CLI

To restore a DB instance from a DB snapshot

- Use the command **rds-restore-db-instance-from-db-snapshot** to restore a DB snapshot to a new DB instance.

```
PROMPT>rds-restore-db-instance-from-db-snapshot mynewdbinstance -s mydbsnap
shot
```

This command returns output similar to the following:

```
DBINSTANCE  mynewdbinstance  db.m1.large  MySQL     50       sa
   creating  3  n  5.1.57  general-public-license
```

API

To restore a DB instance from a DB snapshot

- Call `RestoreDBInstanceFromDBSnapshot` with the following parameters:
  - *`DBSnapshotIdentifier`* = `rds:mysqldb-2014-04-22-08-15`
  - *`DBInstanceIdentifier`* = `mynewdbinstance`

Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=RestoreDBInstanceFromDBSnapshot
   &DBInstanceIdentifier=mynewdbinstance
   &DBSnapshotIdentifier=rds%3Amysqldb-2014-04-22-08-15
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140428/us-east-1/rds/aws4_request
   &X-Amz-Date=20140428T232655Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=78ac761e8c8f54a8c0727f4e67ad0a766fbb0024510b9aa34ea6d1f7df52fe92
```

Related Topics

- Creating a DB Snapshot (p. 499)
- Copying a DB Snapshot (p. 504)

- DB Snapshots (p. 76)

Copying a DB Snapshot

Amazon RDS supports two types of DB snapshot copies. First, you can copy an automated DB snapshot to create a manual DB snapshot in the same AWS region. By creating a manual DB snapshot, the DB snapshot is retained; automated DB snapshots are deleted after their retention period expires.

Second, you can copy either an automated or manual DB snapshot from one region to another region. By copying the DB snapshot to another region, you create a manual DB snapshot that is retained in that region. You perform the DB snapshot copy in the target region, and use an Amazon RDS ARN to specify the location of the DB snapshot in the source region. For information about Amazon RDS ARN formats, see Constructing an Amazon RDS Amazon Resource Name (ARN) (p. 493).

Amazon RDS deletes automated DB snapshots at the end of their retention period, when you disable automated DB snapshots for a DB instance, or when you delete a DB instance. If you want to keep an automated DB snapshot for a longer period, you can copy it, which creates a manual DB snapshot. Manual DB snapshots are retained until you delete them. Note that Amazon RDS storage costs apply to manual DB snapshots.

Copying a DB Snapshot to Another Region

You can have one active cross region DB snapshot copy per destination region per AWS customer account. Copying a snapshot out of the source region incurs Amazon RDS data transfer charges. For more information about Amazon RDS data transfer pricing, go to Amazon Relational Database Service Pricing.

Depending on the regions involved and the amount of data to be copied, a cross region snapshot could take hours to complete. If there are large numbers of cross region DB snapshot copy requests from a given source region, Amazon RDS may queue new cross region copy requests for that source region until some of the in-progress copies have completed. No progress information is displayed about the copy requests while they are in the queue. Progress information is displayed when the copy starts.

After the DB snapshot copy has been created in the new region, the copy behaves the same as all other DB snapshots in that region. For example, the following CLI copy command results in a DB snapshot in the us-west-2 region with the identifier `mysql-instance1-snapshot-20130805-copy`.

```
rds-copy-db-snapshot --source-db-snapshot-identifier arn:aws:rds:us-east-
1:123456789012:snapshot:mysql-instance1-snapshot-20130805 --region us-west-2 -
-target-db-snapshot-identifier mysql-instance1-snapshot-20130805-copy
```

When the copy is finished, the AWS Management Console will show the DB snapshot with the name `mysql-instance1-snapshot-20130805-copy` in your list of DB snapshots in us-west-2. You can perform all DB snapshot actions on the DB snapshot identifier. For example, running the following CLI command in the us-west-2 region will create a new DB instance with data from the DB snapshot copy:

```
rds-restore-db-instance-from-db-snapshot mysql-instance1-west --region us-west-
2 --db-snapshot-identifier mysql-instance1-snapshot-20130805-copy
```

There are some limitations to how and where you can copy DB snapshots. These limitations include:

- You cannot copy a DB snapshot to or from the EU (Frankfurt) or AWS GovCloud (US) region.
- You cannot copy a DB snapshot across regions if it was created from a DB instance that is using Oracle TDE or SQL Server TDE.
- You cannot copy a SQL Server DB snapshot across regions if the DB snapshot was created from an instance using Multi-AZ mirroring.

A snapshot copied across regions does not include either the parameter group or option group that was used by the DB instance the snapshot was created from. When you restore a snapshot to create a new DB instance, that DB instance is assigned the default parameter group and default option group for the region it is created in. To give the new DB instance the same parameters and options as the source, you must do the following:

1. In the destination region, create a parameter group with the same settings as the parameter group used by the source DB instance, or note the name of an existing parameter group that has those settings.
2. In the destination region, create an option group with the same settings as the option group used by the source DB instance, or note the name of an existing option group that has those settings.
3. After restoring the snapshot in the destination region, modify the new DB instance to add the parameter group and option group available in the destination region.

AWS Management Console

To copy a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Snapshots**.
3. Set **Filter:** to **Automated Snapshots**.

   Select the check box for the automated DB snapshot you want to copy.

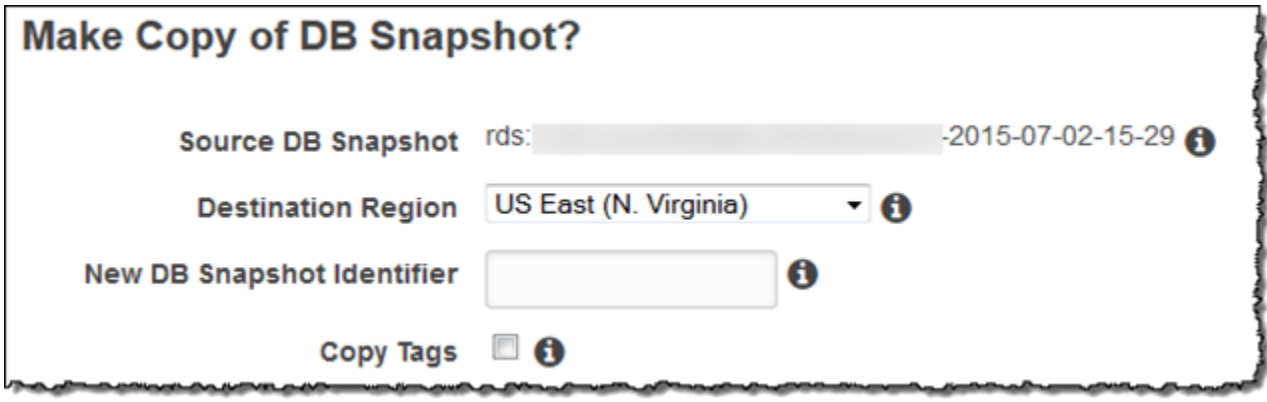
   Click **Copy Snapshot**

   The **Copy DB Snapshot** window appears.
4. Verify that the name of the automated DB snapshot you want to copy appears in the **Source DB Snapshot:** field.

   To copy the DB snapshot to a different region, select that region in the **Destination Region:** list box.

   Type the name of the DB snapshot copy in the **New DB Snapshot Identifier:** text box.

   To copy tags and values from the snapshot to the copy of the snapshot, choose **Copy Tags**.



5. Click **Yes, Copy Snapshot**.

CLI

To copy a DB snapshot

- Use the rds-copy-db-snapshot command to copy a DB snapshot.

  Example

```
rds-copy-db-snapshot -s rds:mydbinstance-2013-09-04-22-50 -t mydbsnapshotcopy
 -ct true
```

  The output from this command should look similar to the following:

```
DBSNAPSHOT  mydbsnapshotcopy  2013-09-04T22:51:29.982Z  mydbinstance  2013-
09-04T22:50:22.355Z  mysql  5  available  MasterUser
default:mysql-5-6  5.6.12  general-public-license  manual
```

API

To copy a DB snapshot

- Call CopyDBSnapshot with the following parameters:
  - *SourceDBSnapshotIdentifier* = arn:aws:rds:us-east-1:815981987263:snapshot:rds:mysqldb-2014-04-27-08-15
  - *TargetDBSnapshotIdentifier* = mydbsnapshotcopy

  Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=CopyDBSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-east-
1%3A815981987263%3Asnapshot%3Ards%3Amydbinstance-2013-09-04-22-50
   &TargetDBSnapshotIdentifier=mydbsnapshotcopy
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-east-1/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Related Topics

- Creating a DB Snapshot (p. 499)
- Restoring From a DB Snapshot (p. 501)
- DB Instance Backups (p. 73)

Restoring a DB Instance to a Specified Time

The Amazon RDS automated backup feature automatically creates a backup of your database. This backup occurs during a daily user-configurable 30 minute period known as the *backup window*. Automated backups are kept for a configurable number of days (called the *backup retention period*). You can restore your DB instance to any specific time during this retention period, creating a new DB instance.

When you restore a DB instance to a point in time, the default DB security group is applied to the new DB instance. If you need custom DB security groups applied to your DB instance, you must apply them explicitly using the AWS Management Console, **ModifyDBInstance** API, or the **rds-modify-db-instance** command line tool once the DB instance is available.

You can restore to any point in time during your backup retention period. To determine the latest restorable time for a DB instance, use the `rds-describe-db-instance` command with the `--show-long` and `--headers` parameters and look at the value returned in the **Latest Restorable Time** column. The latest restorable time for a DB instance is typically within 5 minutes of the current time.

The OFFLINE, EMERGENCY, and SINGLE_USER modes are not currently supported. Setting any database into one of these modes will cause the latest restorable time to stop moving ahead for the whole instance.

Several of the database engines used by Amazon RDS have special considerations when restoring from a point in time. When you restore an Oracle DB instance to a point in time, you can specify a different Oracle DB engine, license model, and DBName (SID) to be used by the new DB instance. When you restore a SQL Server DB instance to a point in time, each database within that instance is restored to a point in time within 1 second of each other database within the instance. Transactions that span multiple databases within the instance may be restored inconsistently.

Some actions, such as changing the recovery model of a SQL Server database, can break the sequence of logs that are use for point-in-time recovery. In some cases, Amazon RDS can detect this issue and the latest restorable time is prevented from moving forward; in other cases, such as when a SQL Server database uses the BULK_LOGGED recovery model, the break in log sequence is not detected. It may not be possible to restore a SQL Server DB instance to a point in time if there is a break in the log sequence. For these reasons, Amazon RDS does not support changing the recovery model of SQL Server databases.

AWS Management Console

To restore a DB instance to a specified time

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **DB Instances**.
3. Click **Instance Actions**, and then click **Restore To Point In Time**.

   The **Restore DB Instance** window appears.
4. Click on the **Use Custom Restore Time** radio button.
5. Enter the date and time that you wish to restore to in the **Use Custom Restore Time** text boxes.
6. Type the name of the restored DB instance in the **DB Instance Identifier** text box.
7. Click the **Launch DB Instance** button.

CLI

To restore a DB instance to a specified time

- Use the command `rds-restore-db-instance-to-point-in-time` to create a new database instance.

```
PROMPT>rds-restore-db-instance-to-point-in-time mytargetdbinstance -s myso
urcedbinstance-db -r 2009-10-14T23:45:00.000Z
```

API

To restore a DB instance to a specified time

- Call `RestoreDBInstanceToPointInTime` with the following parameters:
  - *`SourceDBInstanceIdentifier`* = `mysourcedbinstance`
  - *`TargetDBInstanceIdentifier`* = `mytargetdbinstance`
  - *`RestoreTime`* = `2013-10-14T23:45:00.000Z`

  Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=RestoreDBInstanceToPointInTime
   &RestoreTime=2013-10-14T23%3A45%3A00.000Z
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBInstanceIdentifier=mysourcedbinstance
   &TargetDBInstanceIdentifier=mytargetdbinstance
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-east-1/rds/aws4_request
   &X-Amz-Date=20131016T233051Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=087a8eb41cb1ab0fc9ec1575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Related Topics

- Creating a DB Snapshot (p. 499)
- Restoring From a DB Snapshot (p. 501)
- Copying a DB Snapshot (p. 504)
- DB Instance Backups (p. 73)

Working with Option Groups

Some DB engines offer additional features that make it easier to manage data and databases, and to provide additional security for your database. Amazon RDS uses option groups to enable and configure these features. An option group can specify features, called options, that are available for a particular Amazon RDS DB instance. Options can have settings that specify how the option works. When you associate a DB instance with an option group, the specified options and option settings are enabled for that DB instance.

> **Note**
> Currently, option groups are available for Oracle, Microsoft SQL Server, and MySQL 5.6 DB instances. For more information about individual Oracle options, see Appendix: Options for Oracle Database Engine (p. 226). For more information about SQL Server options, see Appendix: Options for SQL Server Database Engine (p. 350). For more information about MySQL 5.6 options, see Appendix: Options for MySQL Database Engine (p. 173).

Topics

- Option Groups Overview (p. 510)
- Creating an Option Group (p. 514)
- Making a Copy of an Option Group (p. 515)
- Adding an Option to an Option Group (p. 515)
- Listing the Options and Option Settings for an Option Group (p. 518)
- Modifying an Option Setting (p. 519)
- Removing an Option from an Option Group (p. 521)

Option Groups Overview

Amazon RDS provides an empty default option group for each new DB instance. You cannot modify this default option group, but any new option group you create derives its settings from the default option group. To apply an option to a DB instance, you must create an option group (or use an existing option group) and then add one or more options to the option group. You can then associate the DB instance with that option group. To remove all options from a DB instance at once, you associate the DB instance with the default (empty) option group. If you change options or option settings in an option group, those changes are applied to all DB instances that are associated with that option group.

You can copy an existing option group to create a new option group with the rds-copy-option-group command. Copying an option group is a convenient solution when you have already created an option group and you want to include most of the custom parameters and values from that group in a new option group.

When an option group is assigned to a DB instance, it is linked to the supported platform the DB instance is on, either VPC or EC2-Classic (non-VPC). Furthermore, if a DB instance is in a VPC, the option group associated with the instance is linked to that VPC. This means that you cannot use the option group assigned to a DB instance if you attempt to restore the instance into a different VPC or onto a different platform.

If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance. Note that with persistent or permanent options, such as Oracle TDE, you must create a new option group that includes the persistent or permanent option when restoring a DB instance into a different VPC.

Option settings control the behavior of an option. For example, the Oracle Advanced Security option **NATIVE_NETWORK_ENCRYPTION** has a setting that you can use to specify the encryption algorithm

for network traffic to and from the DB instance. Some options settings are optimized for use with Amazon RDS and cannot be changed.

The following options are currently supported on Amazon RDS:

DB Engine	Option ID	Description	Settings
Oracle	`OEM`	Oracle Database Manager Database Control. Default port: 1158.	port value setting
Oracle	`XMLDB`	Oracle XML DB support	No settings
Oracle	`APEX`, `APEX-DEV`	Oracle Application Express (APEX)	No settings
Oracle	`NATIVE_NETWORK _ENCRYPTION`	Oracle native network encryption, a feature of the Oracle Advanced Security option available in Oracle Enterprise Edition.	For a description of all option settings for NATIVE_NETWORK_ENCRYPTION, see Oracle Native Network Encryption.
Oracle	`STATSPACK`	Oracle Statspack performance statistics	No settings
Oracle	`TDE`	Oracle Transparent Data Encryption (TDE)	No settings
Oracle	`TDE_HSM`	Oracle Transparent Data Encryption (TDE)used with Amazon CloudHSM	No settings

DB Engine	Option ID	Description	Settings
Oracle	`Timezone`	Oracle time zone change	The following values are acceptable for the `Timezone` option setting: Africa/Cairo, Africa/Casablanca, Africa/Harare, Africa/Monrovia, Africa/Nairobi, Africa/Tripoli, Africa/Windhoek, America/Araguaina, America/Asuncion, America/Bogota, America/Caracas, America/Chihuahua, America/Cuiaba, America/Denver, America/Fortaleza, America/Guatemala, America/Halifax, America/Manaus, America/Matamoros, America/Monterrey, America/Montevideo, America/Phoenix, America/Santiago, America/Tijuana, Asia/Amman, Asia/Ashgabat, Asia/Baghdad, Asia/Baku, Asia/Bangkok, Asia/Beirut, Asia/Calcutta, Asia/Damascus, Asia/Dhaka, Asia/Irkutsk, Asia/Jerusalem, Asia/Kabul, Asia/Karachi, Asia/Kathmandu, Asia/Krasnoyarsk, Asia/Magadan, Asia/Muscat, Asia/Novosibirsk, Asia/Riyadh, Asia/Seoul, Asia/Shanghai, Asia/Singapore, Asia/Taipei, Asia/Tehran, Asia/Tokyo, Asia/Ulaanbaatar, Asia/Vladivostok, Asia/Yakutsk, Asia/Yerevan, Atlantic/Azores, Australia/Adelaide, Australia/Brisbane, Australia/Darwin, Australia/Hobart, Australia/Perth, Australia/Sydney, Canada/Newfoundland, Canada/Saskatchewan, Europe/Amsterdam, Europe/Athens, Europe/Dublin, Europe/Helsinki, Europe/Istanbul, Europe/Kaliningrad, Europe/Moscow, Europe/Paris, Europe/Prague, Europe/Sarajevo, Pacific/Auckland, Pacific/Fiji, Pacific/Guam, Pacific/Honolulu, Pacific/Samoa, US/Alaska, US/Eastern, US/East-Indiana, US/Pacific, UTC.
MySQL	`MEMCACHED`	MySQL 5.6 memcached interface to InnoDB tables	For a list and description of all the supported memcached parameters, see MySQL 5.6 memcached Support.

DB Engine	Option ID	Description	Settings
SQL Server	`TDE`	SQL Server Transparent Data Encryption	No settings
SQL Server	`Mirroring`	SQL Server implementation of Multi-AZ, where Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone.	No settings

Both DB instances and DB snapshots can be associated with an option group. When you restore from a DB snapshot or when you perform a point-in-time restore for a DB instance, the option group associated with the DB snapshot or DB instance will, by default, be associated with the restored DB instance. You can associate a different option group with a restored DB instance, but the new option group must contain any persistent or permanent options that were included in the original option group.

Option group changes must be applied immediately in two cases: 1) When you add an option that adds or updates a port value, such as the **OEM** option, or 2) when you add or remove an option group with an option that includes a port value. In these cases the `Apply-Immediately` parameter must be enabled. Options that don't include port values can be applied immediately or can be applied during the next maintenance window for the DB instance.

Options require additional memory in order to run on a DB instance, so you may need to launch a larger instance, depending on your current use of the DB instance. For example, Oracle Enterprise Manager Database Control uses about 300 MB of RAM; if you enable this option for a small DB instance, you might encounter performance problems or out-of-memory errors.

Each DB instance indicates the status of its association an option group. For example, a status of **Active** indicates the DB instance is associated with the option group while a status of **Invalid** indicates that the option group associated with the DB instance does not contain the options the DB instance requires. If you query a DB instance for the status of its associated option group, Amazon RDS may also return a `ChangeState` value such as **Pending** or **Applying** when it is attempting to change the association from one state to another. For example, the status of the association of a DB instance in an option group could be **Creating/Pending**.

Persistent and Permanent Options

Two types of options, persistent and permanent, require special consideration when you add them to an option group.

Persistent options, such as the **TDE** option for Microsoft SQL Server transparent data encryption, cannot be removed from an option group while DB instances are associated with the option group. All DB instances must be disassociated from the option group before a persistent option can be removed. When restoring from a snapshot or when performing a point-in-time-restore, if the option group associated with the DB snapshot contains a persistent option, the restored DB instance can only be associated with that option group.

Permanent options, such as the **TDE** option for Oracle Advanced Security TDE, can never be removed from an option group and the option group cannot be disassociated from the DB instance. When restoring from a DB snapshot or when performing a point-in-time-restore, if the option group associated with the snapshot contains a permanent option, the restored DB instance can only be associated with an option group with the permanent option.

Creating an Option Group

You can create a new option group that derives its settings from the default option group, and then add one or more options to the option group. If you already have an existing option group, you can copy that option group with all of the options that you have added into a new option group. You can copy an option group by using the rds-copy-option-group CLI command or the CopyOptionGroup action.

AWS Management Console

To create an option group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Option Groups**.
3. Click **Create Group**.
4. In the **Create Option Group** dialog box, do the following:
   - In the **Name** box, type a name for the option group that is unique within your AWS account. The name can contain only letters, digits, and hyphens.
   - In the **Description** box, type a brief description of the option group. The description will be used for display purposes.
   - In the **Engine** box, click the DB engine that you want.
   - In the **Major Engine Version** box, click the major version of the DB engine that you want.
5. To continue, click **Yes, Create**. To cancel the operation instead, click **Cancel**.

You have now created a new option group with no options. See the next section, Adding an Option to an Option Group (p. 515) to add an option to the option group. Once you have added the options you want, you can then associate a DB instance with the option group so the options become available on the DB instance. For information about associating a DB instance with an option group, see Modifying a DB Instance Running the Oracle Database Engine (p. 216).

CLI

To create an option group

- Call the rds-create-option-group command.

  The following example creates an option group named `TestOptionGroup`, which is associated with the Oracle Enterprise Edition DB engine. The description is enclosed in quotation marks.

```
PROMPT> rds-create-option-group TestOptionGroup --engine-name oracle-ee -
–major-engine-version 11.2 -- description "Test option group"
```

You have now created a new option group with no options. See the next section, Adding an Option to an Option Group (p. 515) to add an option to the option group. Once you have added the options you want, you can then associate a DB instance with the option group so the options become available to the DB instance. For information about associating an Oracle DB instance with an option group, see Modifying a DB Instance Running the Oracle Database Engine (p. 216). For information about associating a MySQL

DB instance with an option group, see Modifying a DB Instance Running the MySQL Database Engine (p. 140).

API

To create an option group

- Call the CreateOptionGroup action.

Making a Copy of an Option Group

You can use the RDS CLI or the RDS API to make a copy of an option group. For example, you can make a copy of an option group that you use in production and then modify the copy to test other option settings.

CLI

To copy an option group

- Call the rds-copy-option-group command.

  The following example creates an option group named `optiongroup2`, which is a copy of the option group `optiongroup1`.

```
PROMPT> rds-copy-option-group optiongroup1 -t optiongroup2 -td "Option group
 2"
```

API

To copy an option group

- Call the CopyOptionGroup action.

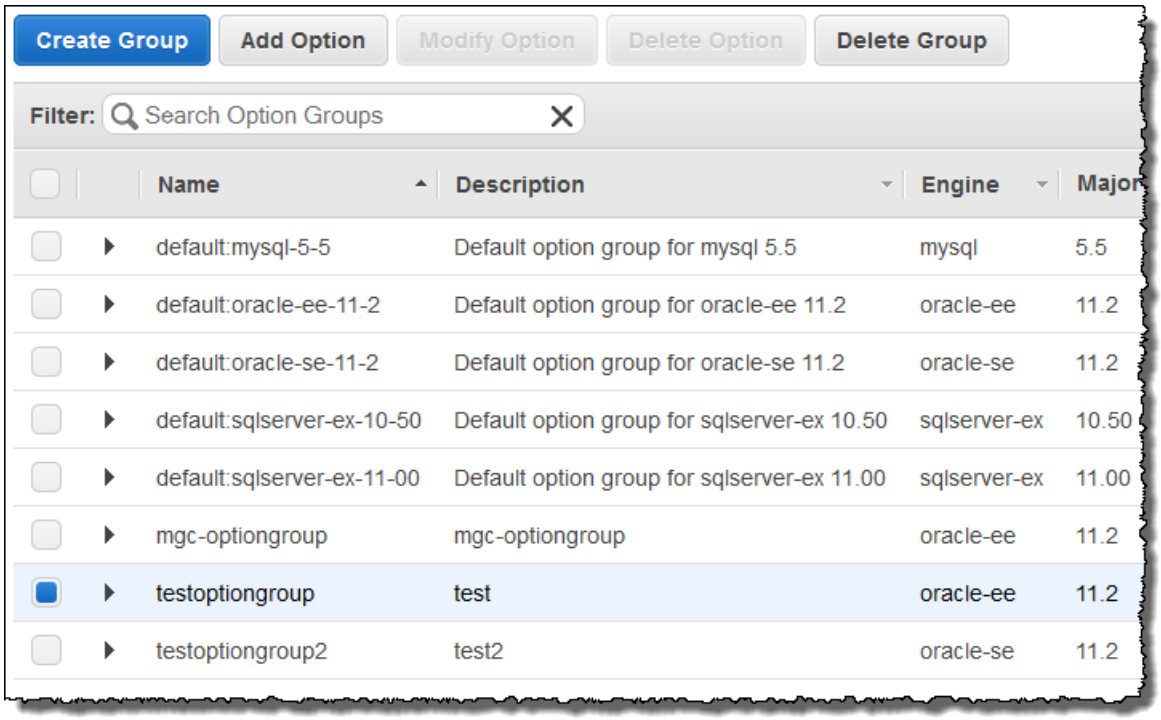
Adding an Option to an Option Group

You can add an option to an existing option group.

AWS Management Console

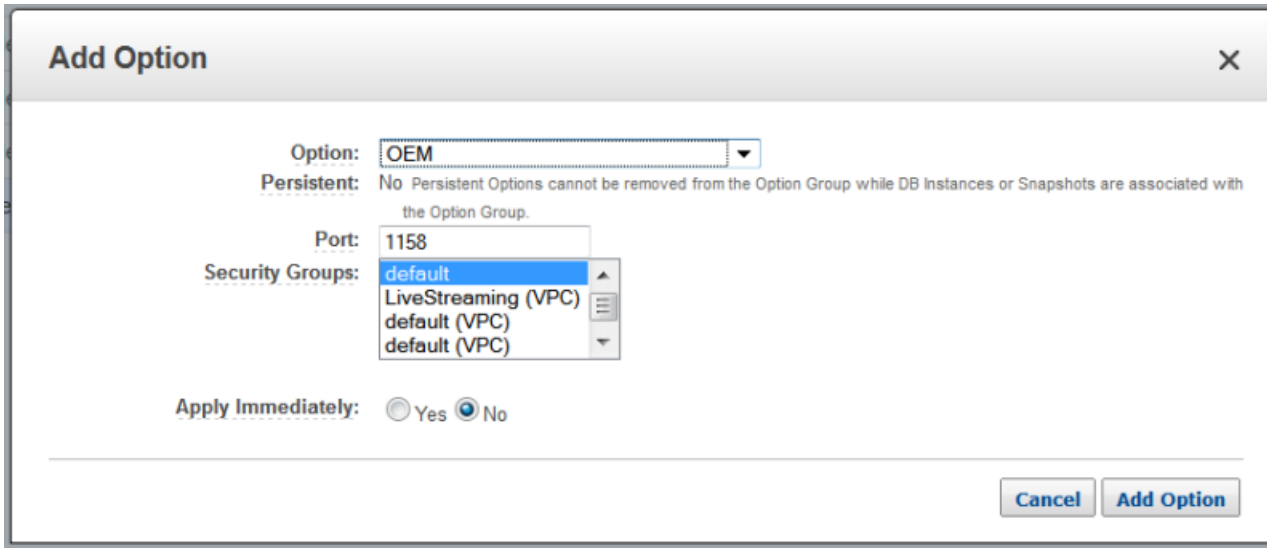
To add an option to an option group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Option Groups**.
3. Select the check box for the option group that you want to modify, and then click **Add Option**.



In the **Add Option** dialog box, do the following:

- Click the option that you want to add. You may need to provide additional values, depending on the option that you select. For example, when you select the OEM option, you must also enter a port value and specify a DB security group.
- To enable the option on all associated DB instances as soon as you add it, under **Apply Immediately**, click **Yes**. If you click **No** (the default), the option will be enabled for each associated DB instance during its next maintenance window.



4. When the settings are as you want them, click **Add Option**.

CLI

To add an option to an option group

- Run the rds-add-option-to-option-group command with the option that you want to add. To enable the new option immediately on all associated DB instances, include the `--apply-immediately` argument. By default, the option will be enabled for each associated DB instance during its next maintenance window.

  The following example adds the Oracle Enterprise Manager Database Control (OEM) to an option group named `TestOptionGroup` and immediately enables it. Note that even if you use the default security group, you must specify it.

```
 PROMPT> rds-add-option-to-option-group TestOptionGroup –-option-name OEM
--security-groups default --apply-immediately
```

  Command output is similar to the following:

```
OPTIONGROUP testoptiongroup oracle-ee 11.2 Test option group
    OPTION OEM 1158 Oracle Enterprise Manager
        SECGROUP default authorized
```

  The following example adds the Oracle OEM option to an option group, specifies a custom port, and specifies a pair of Amazon EC2 VPC security groups to use for that port.

```
PROMPT> rds-add-option-to-option-group my-option-group -n OEM --port 5432
-vpcsg sg-454fa22a,sg-5da54932
```

  Command output is similar to the following:

```
OPTIONGROUP  my-option-group  oracle-se  11.2  My option group
OPTION  OEM  n  5432  Oracle Enterprise Manager
VPCSECGROUP  sg-454fa22a  active
VPCSECGROUP  sg-5da54932  active
```

  The following example adds the Oracle option NATIVE_NETWORK_ENCRYPTION to an option group and specifies the option settings. If no option settings are specified, default values are used.

```
PROMPT> rds-add-option-to-option-group my-option-group -n NATIVE_NETWORK_EN
CRYPTION --settings "SQLNET.ENCRYPTION_SERVER=REQUIRED; SQLNET.ENCRYP
TION_TYPES_SERVER=AES256,AES192,DES"
```

  Command output is similar to the following:

```
OPTIONGROUP  Group Name       Engine     Major Engine Version  Description
      VpcSpecific
OPTIONGROUP  my-option-group  oracle-ee  11.2                  My option
group  n
    OPTION  Name                          Persistent Permanent Description
    OPTION  NATIVE_NETWORK_ENCRYPTION   n          n         Oracle Advanced
 Security - Native Network Encryption
      OPTIONSETTING  Name                                    Description
                                                 Value
Modifiable
      OPTIONSETTING  SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER  Specifies list of
 checksumming algorithms in order of intended use  SHA1,MD5           true
      OPTIONSETTING  SQLNET.ENCRYPTION_TYPES_SERVER       Specifies list of
 encryption algorithms in order of intended use    AES256,AES192,DES  true
      OPTIONSETTING  SQLNET.ENCRYPTION_SERVER             Specifies the de
sired encryption behavior                              REQUIRED           true

      OPTIONSETTING  SQLNET.CRYPTO_CHECKSUM_SERVER        Specifies the de
sired data integrity behavior                          REQUESTED          true
```

API

To add an option to an option group

- Call the ModifyOptionGroup action.

Listing the Options and Option Settings for an Option Group

You can list all the options and option settings for an option group.

AWS Management Console

To list the options and option settings for an option group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Option Groups**. The **Options** column in the table shows the options and option settings in the option group.

CLI

To list the options and option settings for an option group

- Run the rds-describe-option-groups command. Specify the name of the option group whose options and settings you want to view. If you do not specify an option group name, all option groups are described.

The following example lists the options and option settings for an option group named `TestOptionGroup`.

```
PROMPT>rds-describe-option-groups TestOptionGroup
```

API

To list the options and option settings for an option group

- Call the DescribeOptionGroups action.

Modifying an Option Setting

After you have added an option that has modifiable option settings, you can modify the settings at any time.

AWS Management Console

To modify an option setting

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Option Groups**.
3. Click the check box for the option group whose option that you want to modify, and then click **Modify Option**.
4. In the **Modify Option** dialog box, in the **Installed Options** box, click the option whose setting you want to modify. Make the changes that you want.
5. To enable the option as soon as you add it, under **Apply Immediately** , click **Yes**. If you click **No** (the default), the option will be enabled for each associated DB instance during its next maintenance window.
6. When the settings are as you want them, click **Modify Option**.

CLI

To modify an option setting

- Run the rds-add-option-to-option-group command with the option group and option that you want to modify. To apply the change immediately to all associated DB instances, include the `--apply-immediately` argument. By default, the option will be enabled for each associated DB instance during its next maintenance window.

  To set an option setting, use the `--settings` argument. For more information on what settings are available for the various options, see Option Groups Overview (p. 510)

  The following example modifies the port that the Oracle Enterprise Manager Database Control (OEM) uses in an option group named `TestOptionGroup` and immediately applies the change.

```
PROMPT> rds-add-option-to-option-group TestOptionGroup -–option-name OEM -
-port 5432 --apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP testoptiongroup oracle-ee 11.2 Test Option Group
    OPTION OEM 5432 Oracle Enterprise Manager
        SECGROUP default authorized
```

The following example modifies the Oracle option NATIVE_NETWORK_ENCRYPTION and changes the option settings.

```
PROMPT> rds-add-option-to-option-group my-option-group -n NATIVE_NETWORK_EN
CRYPTION --settings "SQLNET.ENCRYPTION_SERVER=REQUIRED; SQLNET.ENCRYP
TION_TYPES_SERVER=AES256,AES192,DES"
```

Command output is similar to the following:

```
OPTIONGROUP  Group Name       Engine     Major Engine Version  Description
      VpcSpecific
OPTIONGROUP  my-option-group  oracle-ee  11.2                  My option
group  n
    OPTION  Name                         Persistent Permanent Description
    OPTION  NATIVE_NETWORK_ENCRYPTION   n          n         Oracle Advanced
 Security - Native Network Encryption
      OPTIONSETTING  Name                                  Description
                                                  Value
Modifiable
      OPTIONSETTING  SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER  Specifies list of
 checksumming algorithms in order of intended use  SHA1,MD5           true
      OPTIONSETTING  SQLNET.ENCRYPTION_TYPES_SERVER       Specifies list of
 encryption algorithms in order of intended use    AES256,AES192,DES  true
      OPTIONSETTING  SQLNET.ENCRYPTION_SERVER             Specifies the de
sired encryption behavior                          REQUIRED           true

      OPTIONSETTING  SQLNET.CRYPTO_CHECKSUM_SERVER        Specifies the de
sired data integrity behavior                      REQUESTED          true
```

API

To modify an option setting

- Call ModifyOptionGroup action.

Removing an Option from an Option Group

You can remove an option from an option group. Even if you remove all options from an option group, Amazon RDS will not delete it. DB instances that are associated with the empty option group will continue to be associated with it; they just won't have access to any options.

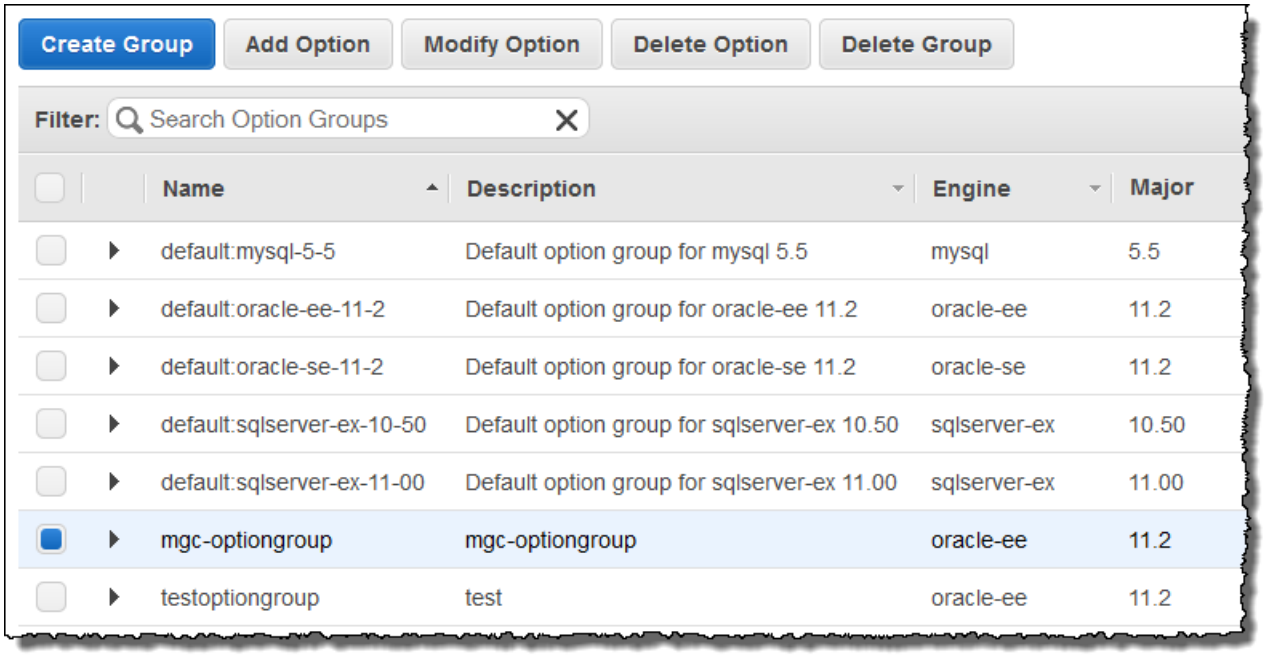
You can remove an option from an option group as long as the option is not persistent or permanent. A persistent option cannot be removed from an option group until all DB instances associated with that option group are disassociated. A permanent option can never be removed from an option group.

Even if you remove all options from an option group, Amazon RDS will not delete it. DB instances that are associated with the empty option group will continue to be associated with it; they just won't have access to any options.

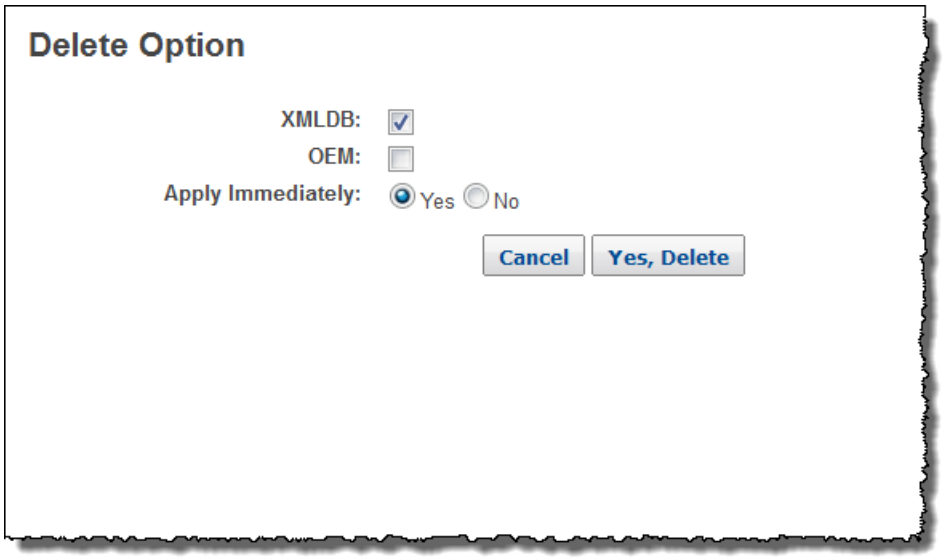
AWS Management Console

To remove an option from an option group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Option Groups**.
3. Select the check box for the option group whose option you want to remove, and then click **Delete Option**.



4. In the **Delete Option** dialog box, do the following:
   - Select the check box that corresponds to the option that you want to delete.
   - For the deletion to take effect as soon as you make it, under **Apply Immediately**, click **Yes**. If you select **No** (the default), the option will be deleted for each associated DB instance during its next maintenance window.



5. When the settings are as you want them, click **Yes, Delete**.

CLI

To remove an option from an option group

- Run the rds-remove-option-from-option-group command with the option that you want to delete. To apply the change immediately, include the `--apply-immediately` argument. By default, the option is deleted on each associated DB instance during its next maintenance window.

  The following example removes the Oracle Enterprise Manager Database Control (OEM) option from an option group named `TestOptionGroup` and immediately applies the change.

```
PROMPT> rds-remove-option-from-option-group TestOptionGroup –options OEM --apply-immediately
```

  Command output is similar to the following:

```
OPTIONGROUP    testoptiongroup oracle-ee   11.2    Test option group
```

API

To remove an option from an option group

- Call the ModifyOptionGroup action.

Working with DB Parameter Groups

You manage your DB engine configuration through the use of parameters in a DB parameter group. DB parameter groups act as a *container* for engine configuration values that are applied to one or more DB instances.

A default DB parameter group is created if you create a DB instance without specifying a customer-created DB parameter group. This default group contains database engine defaults and Amazon RDS system defaults based on the engine, compute class, and allocated storage of the instance. You cannot modify the parameter settings of a default DB parameter group; you must create your own DB parameter group to change parameter settings from their default value. Note that not all DB engine parameters can be changed in a customer-created DB parameter group.

If you want to use your own DB parameter group, you simply create a new DB parameter group, modify the desired parameters, and modify your DB instance to use the new DB parameter group. All DB instances that are associated with a particular DB parameter group get all parameter updates to that DB parameter group. You can also copy an existing parameter group with the rds-copy-db-parameter-group command. Copying a parameter group is a convenient solution when you have already created a DB parameter group and you want to include most of the custom parameters and values from that group in a new DB parameter group.

Here are some important points you should know about working with parameters in a DB parameter group:

- When you change a dynamic parameter and save the DB parameter group, the change is applied immediately regardless of the **Apply Immediately** setting. When you change a static parameter and save the DB parameter group, the parameter change will take effect after you manually reboot the DB instance. You can reboot a DB instance using the RDS console or explicitly calling the `RebootDbInstance` API action (without failover, if the DB instance is in a Multi-AZ deployment). The requirement to reboot the associated DB instance after a static parameter change helps mitigate the risk of a parameter misconfiguration affecting an API call, such as calling `ModifyDBInstance` to change DB instance class or scale storage.
- When you change the DB parameter group associated with a DB instance, you must manually reboot the instance before the new DB parameter group is used by the DB instance.
- The value for a DB parameter can be specified as an integer, or an integer expression built from formulas, variables, functions, and operators. For more information, see DB Parameter Values (p. 534)
- Improperly setting parameters in a DB parameter group can have unintended adverse effects, including degraded performance and system instability. Always exercise caution when modifying database parameters and back up your data before modifying a DB parameter group. You should try out parameter group setting changes on a test DB instance before applying those parameter group changes to a production DB instance.
- Amazon Aurora uses both DB parameter groups and DB cluster parameter groups. Parameters in a DB parameter group apply to a single DB instance in an Aurora DB cluster. Parameters in a DB cluster parameter group apply to every DB instance in a DB cluster. For more information, see Appendix: DB Cluster and DB Instance Parameters (p. 437).

Topics

- Creating a DB Parameter Group (p. 524)
- Modifying Parameters in a DB Parameter Group (p. 525)
- Copying a DB Parameter Group (p. 528)
- Listing DB Parameter Groups (p. 529)
- Viewing Parameter Values for a DB Parameter Group (p. 531)
- DB Parameter Values (p. 534)

Creating a DB Parameter Group

The following section shows you how to create a new DB parameter group.

Important
After you create a DB parameter group, you should wait at least 5 minutes before creating your first DB instance that uses that DB parameter group as the default parameter group. This allows Amazon RDS to fully complete the create action before the parameter group is used as the default for a new DB instance. This is especially important for parameters that are critical when creating the default database for a DB instance, such as the character set for the default database defined by the `character_set_database` parameter. You can use the Parameter Groups option of the Amazon RDS console or the rds-describe-db-parameters command to verify that your DB parameter group has been created or modified.

AWS Management Console

To create a DB parameter group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Click **Parameter Groups** in the **Navigation** list on the left side of the window.
3. Click the **Create DB Parameter Group** button.

   The **Create DB Parameter Group** window appears.
4. Select a DB parameter group family in the **DB Parameter Group Family** drop-down list box.
5. Type the name of the new DB parameter group in the **DB Parameter Group** text box.
6. Type a description for the new DB parameter group in the **Description** text box.
7. Click the **Yes, Create** button.

CLI

To create a DB parameter group

- Use the rds-create-db-parameter-group command. The following example creates a DB parameter group named *mydbparametergroup* for MySQL version 5.1 with a description of "*My new parameter group*."

```
PROMPT>rds-create-db-parameter-group mydbparametergroup -f MySQL5.1 -d "My
 new parameter group"
```

This command produces output similar to the following:

```
DBPARAMETERGROUP  mydbparametergroup  mysql5.1  My new parameter group
```

API

To create a DB parameter group

- Call CreateDBParameterGroup. The following examples creates a new parameter group named *mydbparametergroup* for MySQL version 5.1 and that has the description "*My new parameter group*."

  Example

```
https://rds.amazonaws.com/
 ?Action=CreateDBParameterGroup
 &DBParameterGroupName=mydbparametergroup
 &Description=My%20new%20parameter%20group
 &DBParameterGroupFamily=MySQL5.1
 &Version=2012-01-15
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2012-01-15T22%3A06%3A23.624Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

  The command returns a response like the following:

```
<CreateDBParameterGroupResponse xmlns="http://rds.amazonaws.com/admin/2012-
01-15/">
  <CreateDBParameterGroupResult>
    <DBParameterGroup>
      <DBParameterGroupFamily>mysql5.1</DBParameterGroupFamily>
      <Description>My new parameter group</Description>
      <DBParameterGroupName>mydbparametergroup</DBParameterGroupName>
    </DBParameterGroup>
  </CreateDBParameterGroupResult>
  <ResponseMetadata>
    <RequestId>700a8afe-0b81-11df-85f9-eb5c71b54ddc</RequestId>
  </ResponseMetadata>
</CreateDBParameterGroupResponse>
```

Modifying Parameters in a DB Parameter Group

You can modify parameter values in a customer-created DB parameter group; you cannot change the parameter values in a default DB parameter group. Changes to parameters in a customer-created DB parameter group are applied to all DB instances that are associated with the DB parameter group.

If you change a parameter value, when the change is applied is determined by the type of parameter. Changes to dynamic parameters are applied immediately. Changes to static parameters require that the DB instance associated with DB parameter group be rebooted before the change takes effect. To determine the type of a parameter, list the parameters in a parameter group using one of the procedures shown in the section Listing DB Parameter Groups (p. 529).

After you modify a DB parameter group, you should wait at least 5 minutes before creating your first DB instance that uses that DB parameter group as the default parameter group. This allows Amazon RDS to fully complete the modify action before the parameter group is used as the default for a new DB instance. This is especially important for parameters that are critical when creating the default database for a DB instance, such as the character set for the default database defined by the `character_set_database`

parameter. You can use the Parameter Groups option of the Amazon RDS console or the rds-describe-db-parameters command to verify that your DB parameter group has been created or modified.

The RDS console shows the status of the DB parameter group associated with a DB instance. For example, if the DB instance is not using the latest changes to its associated DB parameter group, the RDS console shows the DB parameter group with a status of **pending-reboot**. You would need to manually reboot the DB instance for the latest parameter changes to take effect for that DB instance.

AWS Management Console

To modify a DB parameter group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Click **Parameter Groups** in the navigation pane on the left side of the window.

   The available DB parameter groups appear in a list.
3. In the list, select the parameter group you want to modify.
4. Select **Edit Parameters**.
5. Change the values of the parameters you want to modify. You can scroll through the parameters using the arrow keys at the top right of the dialog box.

   Note that you cannot change values in a default parameter group.
6. Click **Save Changes**.

CLI

To modify a DB parameter group

- Use the `rds-modify-db-parameter-group` command. The following example modifies the `max_connections` and `max_allowed_packet` values in the DB parameter group named *mydbparametergroup*.

  Note
  Amazon RDS does not support passing multiple comma-delimited parameter values for a single parameter.

```
PROMPT>rds-modify-db-parameter-group mydbparametergroup --parameters
"name=max_connections,value=250,method=immediate" --parameters
"name=max_allowed_packet,value=1024,method=immediate"
```

The command produces output like the following:

```
DBPARAMETERGROUP  mydbparametergroup
```

API

To modify a DB parameter group

Note
Amazon RDS does not support passing multiple comma-delimited parameter values for a single parameter.

- Call the ModifyDBParameterGroup action. The following example modifies the `max_connections` and `max_allowed_packet` values in the DB parameter group named *mydbparametergroup*.

Example

```
https://rds.amazonaws.com/
   ?Action=ModifyDBParameterGroup
   &DBParameterGroupName=mydbparametergroup
   &Parameters.member.1.ParameterName=max_allowed_packet
   &Parameters.member.1.ParameterValue=1024
   &Parameters.member.1.ApplyMethod=immediate
   &Version=2012-01-15
   &SignatureVersion=2
   &SignatureMethod=HmacSHA256
   &Timestamp=2012-01-15T22%3A29%3A47.865Z
```

The command returns a response like the following:

```
<ModifyDBParameterGroupResponse xmlns="http://rds.amazonaws.com/admin/2012-
01-15/">
  <ModifyDBParameterGroupResult>
    <DBParameterGroupName>mydbparametergroup</DBParameterGroupName>
  </ModifyDBParameterGroupResult>
  <ResponseMetadata>
    <RequestId>3b824e10-0b87-11df-972f-21e99bc6881d</RequestId>
  </ResponseMetadata>
</ModifyDBParameterGroupResponse>
```

Copying a DB Parameter Group

You can copy custom DB parameter groups that you create. Copying a parameter group is a convenient solution when you have already created a DB parameter group and you want to include most of the custom parameters and values from that group in a new DB parameter group. You can copy a DB parameter group by using the rds-copy-db-parameter-group CLI command or the CopyDBParameterGroup action.

After you copy a DB parameter group, you should wait at least 5 minutes before creating your first DB instance that uses that DB parameter group as the default parameter group. This allows Amazon RDS to fully complete the copy action before the parameter group is used as the default for a new DB instance. This is especially important for parameters that are critical when creating the default database for a DB instance, such as the character set for the default database defined by the `character_set_database` parameter. You can use the Parameter Groups option of the Amazon RDS console or the rds-describe-db-parameters command to verify that your DB parameter group has been created.

CLI

To copy a DB parameter group

- Use the `rds-copy-db-parameter-group` command. The following example creates a new DB parameter group named `mygroup2` that is a copy of the DB parameter group `mygroup1`.

```
PROMPT> rds-copy-db-parameter-group mygroup1 -t mygroup2 -td "DB parameter
 group 2"
```

API

To copy a DB parameter group

- Call the CopyDBParameterGroup action. The following example creates a new DB parameter group named `mygroup2` that is a copy of the DB parameter group `mygroup1`.

Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=CopyDBParameterGroup
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBParameterGroupIdentifier=arn%3Aaws%3Ards%3Aus-west-
2%3A815981987263%3pg%3Amygroup1
   &TargetDBParameterGroupIdentifier=mygroup2
   &TargetDBParameterGroupDescription=DB%20parameter%20group%202
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140922/us-east-1/rds/aws4_request
   &X-Amz-Date=20140922T175351Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=5164017efa99caf850e874a1cb7ef62f3ddd29d0b448b9e0e7c53b288ddffed2
```

The command returns a response like the following:

```
<CopyDBParameterGroupResponse xmlns="http://rds.amazonaws.com/doc/2014-09-
01/">
  <CopyDBParameterGroupResult>
    <DBParameterGroup>
      <DBParameterGroupFamily>mysql5.6</DBParameterGroupFamily>
      <Description>DB parameter group 2</Description>
      <DBParameterGroupName>mygroup2</DBParameterGroupName>
    </DBParameterGroup>
  </CopyDBParameterGroupResult>
  <ResponseMetadata>
    <RequestId>3328d60e-beb6-11d3-8e5c-3ccda5460d76</RequestId>
  </ResponseMetadata>
</CopyDBParameterGroupResponse>
```

Listing DB Parameter Groups

You can list the DB parameter groups you've created for your AWS account.

Note
Default parameter groups are automatically created from a default parameter template when you create a DB instance for a particular DB engine and version. These default parameter groups contain preferred parameter settings and cannot be modified. When you create a custom parameter group, you can modify parameter settings.

AWS Management Console

To list all DB parameter groups for an AWS account

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Click **Parameter Groups** in the navigation pane on the left side of the window.

   The DB parameter groups appear in a list.

CLI

To list all DB parameter groups for an AWS account

- Use the command rds-describe-db-parameter-groups command. The following example lists all available DB parameter groups for an AWS account.

```
PROMPT>rds-describe-db-parameter-groups
```

The command returns a response like the following:

```
DBPARAMETERGROUP  default.mysql5.1     mysql5.1  Default parameter group
for MySQL5.1
DBPARAMETERGROUP  default.mysql5.5     mysql5.5  Default parameter group
for MySQL5.5
DBPARAMETERGROUP  mydbparametergroup   mysql5.5  My new parameter group
```

API

To list all DB parameter groups for an AWS account

- Call DescribeDBParameterGroups action. The following example returns a list of DB parameter groups.

Example

```
https://rds.amazonaws.com/
 ?Action=DescribeDBParameterGroups
 &MaxRecords=100
 &Version=2012-01-15
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-22T19%3A31%3A42.262Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

The command returns a response like the following:

```
<DescribeDBParameterGroupsResponse xmlns="http://rds.amazonaws.com/admin/2012-
01-15/">
    <DescribeDBParameterGroupsResult>
        <DBParameterGroups>
            <DBParameterGroup>
                <Engine>mysql5.1</Engine>
                <Description>Default parameter group for MySQL5.1</Descrip
tion>
              <DBParameterGroupName>default.mysql5.1</DBParameterGroupName>

            </DBParameterGroup>
            <DBParameterGroup>
                <Engine>mysql5.1</Engine>
                <Description>My new parameter group</Description>
               <DBParameterGroupName>mydbparametergroup</DBParameterGroup
Name>
            </DBParameterGroup>
        </DBParameterGroups>
    </DescribeDBParameterGroupsResult>
    <ResponseMetadata>
        <RequestId>41731881-0b82-11df-9a9b-c1bd5894571c</RequestId>
    </ResponseMetadata>
</DescribeDBParameterGroupsResponse>
```

Viewing Parameter Values for a DB Parameter Group

You can get a list of all parameters in a DB parameter group and their values.

AWS Management Console

To view the parameter values for a DB parameter group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.

2. Click **Parameter Groups** in the navigation pane on the left side of the window.

   The DB parameter groups appear in a list.
3. Select a DB parameter group from the list. Click the Details page icon to see the list of parameters for the selected DB parameter group.

CLI

To view the parameter values for a DB parameter group

- Use the rds-describe-db-parameters command. The following example lists the parameters and parameter values for a DB parameter group named mydbparametergroup.

```
PROMPT>rds-describe-db-parameters mydbparametergroup
```

The command returns a response like the following:

```
DBPARAMETER  Parameter Name            Parameter Value  Source
Data Type  Apply Type  Is Modifiable
DBPARAMETER  allow-suspicious-udfs                      engine-default
boolean    static      false
DBPARAMETER  auto_increment_increment                   engine-default
integer    dynamic     true
DBPARAMETER  auto_increment_offset                      engine-default
integer    dynamic     true
DBPARAMETER  binlog_cache_size         32768            system
integer    dynamic     true
DBPARAMETER  socket                    /tmp/mysql.sock  system
string     static      false
```

API

To view the parameter values for a DB Parameter Group

- Call DescribeDBParameters action. The following example lists the parameters and parameter values for a DB parameter group named mydbparametergroup.

Example

```
https://rds.amazonaws.com/
 ?Action=DescribeDBParameters
 &DBParameterGroupName=mydbparametergroup
 &MaxRecords=100
 &Version=2012-01-15
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-22T19%3A31%3A42.262Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

The command returns a response like the following:

```
<DescribeDBParametersResponse xmlns="http://rds.amazonaws.com/admin/2012-
01-15/">
  <DescribeDBParametersResult>
    <Marker>bWF4X3RtcF90YWJsZXM=</Marker>
    <Parameters>
      <Parameter>
        <DataType>boolean</DataType>
        <Source>engine-default</Source>
        <IsModifiable>false</IsModifiable>
       <Description>Controls whether user-defined functions that have only
 an xxx symbol for the main function can be loaded</Description>
        <ApplyType>static</ApplyType>
        <AllowedValues>0,1</AllowedValues>
        <ParameterName>allow-suspicious-udfs</ParameterName>
      </Parameter>
      <Parameter>
        <DataType>integer</DataType>
        <Source>engine-default</Source>
        <IsModifiable>true</IsModifiable>
        <Description>Intended for use with master-to-master replication,
and can be used to control the operation of AUTO_INCREMENT columns</Descrip
tion>
        <ApplyType>dynamic</ApplyType>
        <AllowedValues>1-65535</AllowedValues>
        <ParameterName>auto_increment_increment</ParameterName>
      </Parameter>
      <Parameter>
        <DataType>integer</DataType>
        <Source>engine-default</Source>
        <IsModifiable>true</IsModifiable>
        <Description>Determines the starting point for the AUTO_INCREMENT
column value</Description>
        <ApplyType>dynamic</ApplyType>
        <AllowedValues>1-65535</AllowedValues>
        <ParameterName>auto_increment_offset</ParameterName>
      </Parameter>

      (... sample truncated...)
```

```
    </Parameters>
  </DescribeDBParametersResult>
  <ResponseMetadata>
    <RequestId>99c0937a-0b83-11df-85f9-eb5c71b54ddc</RequestId>
  </ResponseMetadata>
</DescribeDBParametersResponse>
```

DB Parameter Values

The value for a DB parameter can be specified as:

- An integer constant.
- A DB parameter formula.
- A DB parameter function.
- A character string constant.

DB Parameter Formulas

A DB parameter formula is an expression that resolves to an integer value, and is enclosed in braces: {}. Formulas can be specified for either a DB parameter value or as an argument to a DB parameter function.

Syntax

```
{FormulaVariable}
```

```
{FormulaVariable*Integer}
```

```
{FormulaVariable*Integer/Integer}
```

```
{FormulaVariable/Integer}
```

DB Parameter Formula Variables

Formula variables return integers. The names of the variables are case sensitive.

AllocatedStorage
: Returns the size, in bytes, of the data volume.

DBInstanceClassMemory
: Returns the number of bytes of memory allocated to the DB instance class associated with the current DB instance, less the memory used by the Amazon RDS processes that manage the instance.

EndPointPort
: Returns the number of the port used when connecting to the DB instance.

DB Parameter Formula Operators

DB parameter formulas support two operators: division and multiplication.

Division Operator: /
: Divides the dividend by the divisor, returning an integer quotient. Decimals in the quotient are truncated, not rounded.

  Syntax

```
dividend / divisor
```

  The dividend and divisor arguments must be integer expressions.

Multiplication Operator: *
: Divides the dividend by the divisor, returning an integer quotient. Decimals in the quotient are truncated, not rounded.

  Syntax

```
expression * expression
```

  Both expressions must be integers.

DB Parameter Functions

The parameter arguments can be specified as either integers or formulas. Each function must have at least one argument. Multiple arguments can be specified as a comma-separated list. The list cannot have any empty members, such as *argument1*,,*argument3*. Function names are case insensitive.

Note
DB Parameter functions are not currently supported in CLI.

GREATEST()
: Returns the largest value from a list of integers or parameter formulas.

  Syntax

```
GREATEST(argument1, argument2,...argumentn)
```

  Returns an integer.

LEAST()
: Returns the smallest value from a list of integers or parameter formulas.

  Syntax

```
LEAST(argument1, argument2,...argumentn)
```

  Returns an integer.

SUM()
: Adds the values of the specified integers or parameter formulas.

  Syntax

```
SUM(argument1, argument2,...argumentn)
```

  Returns an integer.

DB Parameter Value Examples

These examples show using formulas and functions in the values for DB parameters.

Caution
Improperly setting parameters in a DB parameter group can have unintended adverse effects, including degraded performance and system instability. Always exercise caution when modifying database parameters and back up your data before modifying your DB parameter group. You should try out parameter group changes on a test DB instances, created using point-in-time-restores, before applying those parameter group changes to your production DB instances.

You can specify the GREATEST function in an Oracle processes parameter to set the number of user processes to the larger of either 80 or DBInstanceClassMemory divided by 9868951.

```
GREATEST({DBInstanceClassMemory/9868951},80)
```

You can specify the LEAST() function in a MySQL max_binlog_cache_size parameter value to set the maximum cache size a transaction can use in a MySQL instance to the lesser of 1MB or DBInstanceClass/256:

```
LEAST({DBInstanceClassMemory/256},10485760)
```

Working with DB Security Groups

A DB security group controls network access to a DB instance that is not inside a VPC. By default, network access is turned off to a DB instance. You can specify rules in a security group that allows access from an IP address range, port, or EC2 security group. Once ingress rules are configured, the same rules apply to all DB instances that are associated with that security group. You can specify up to 20 rules in a security group.

If you are a new customer to Amazon RDS or if you are an existing customer who is using a new region, your DB instance is most likely in a default VPC. You cannot use a DB security group for a DB instance inside a VPC; you must create a VPC security group. For information on creating a VPC security group, see Security Groups for Your VPC. To determine if you have a default VPC, see step 2 in the following procedure.

Topics

- Creating a DB Security Group (p. 537)
- Listing Available DB Security Groups (p. 540)
- Viewing a DB security group (p. 541)
- Authorizing Network Access to a DB Security Group from an IP Range (p. 543)
- Authorizing Network Access to a DB Instance from an Amazon EC2 Instance (p. 545)
- Revoking Network Access to a DB Instance from an IP Range (p. 547)
- Related Topics (p. 549)

Creating a DB Security Group

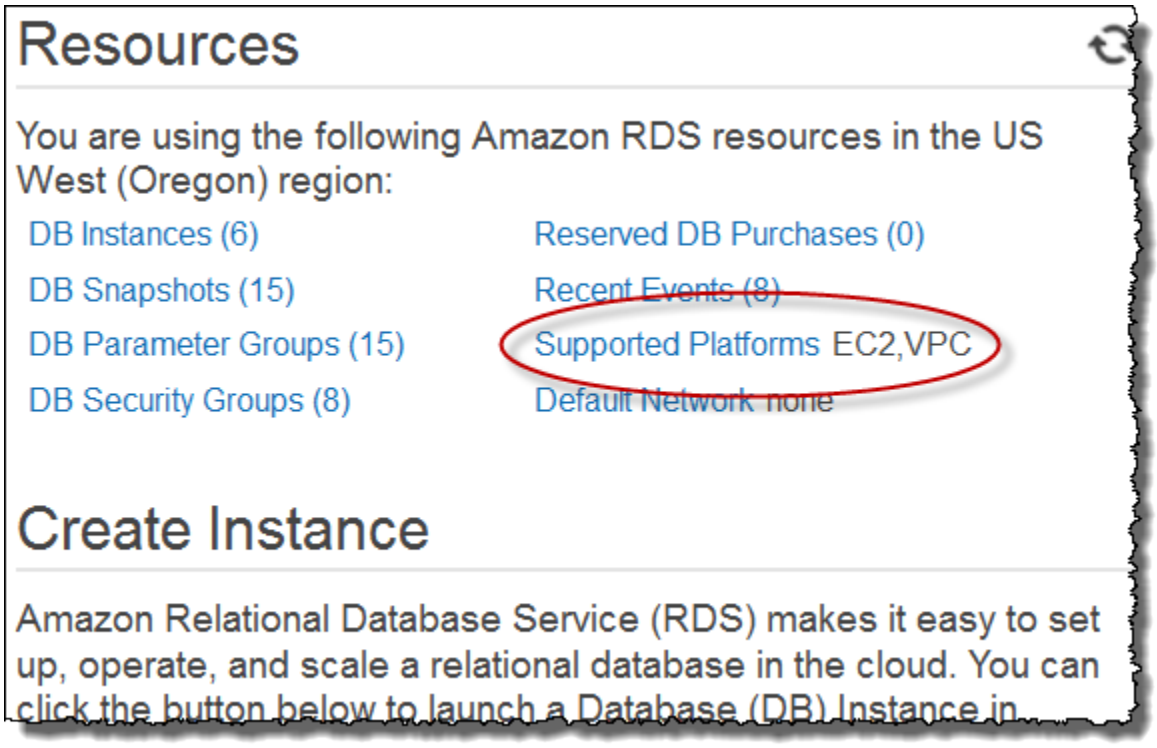
To create a DB security group, you need to provide a name and a description.

AWS Management Console

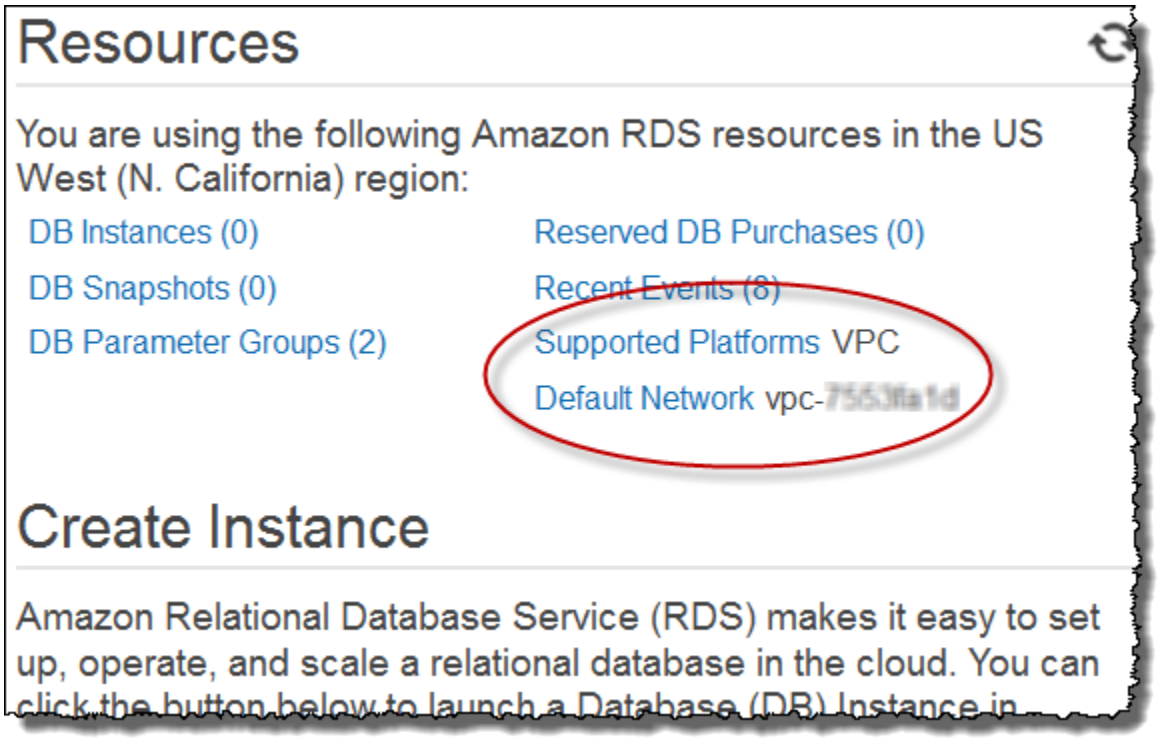
To create a DB security group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Determine what platforms are supported for your AWS account in your current region.

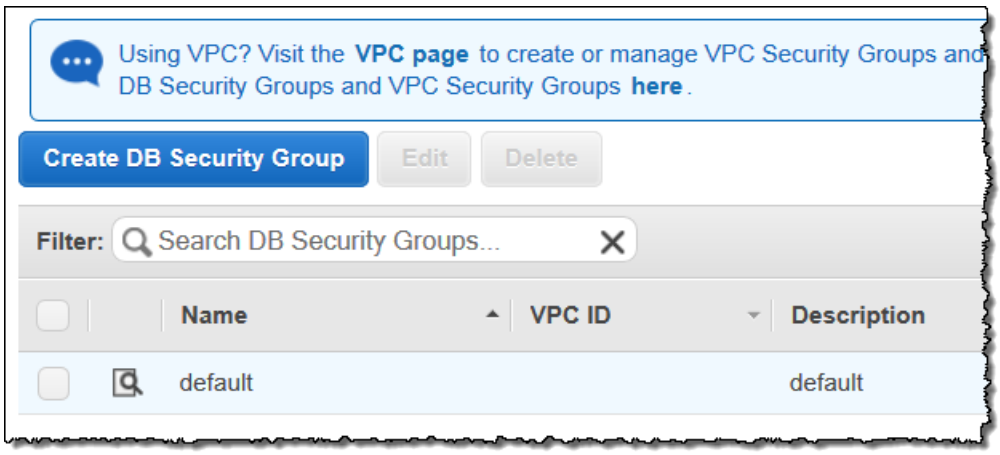
   If **Supported Platforms** indicates *EC2, VPC*, your AWS account in the current region does not use a default VPC. You can continue following the steps below to create a DB security group that will enable access to your DB instance.


Resources
You are using the following Amazon RDS resources in the US West (Oregon) region:
DB Instances (6)
Reserved DB Purchases (0)
DB Snapshots (15)
Recent Events (8)
DB Parameter Groups (15)
Supported Platforms EC2,VPC
DB Security Groups (8)
Default Network none
Create Instance
Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud. You can click the button below to launch a Database (DB) Instance in...

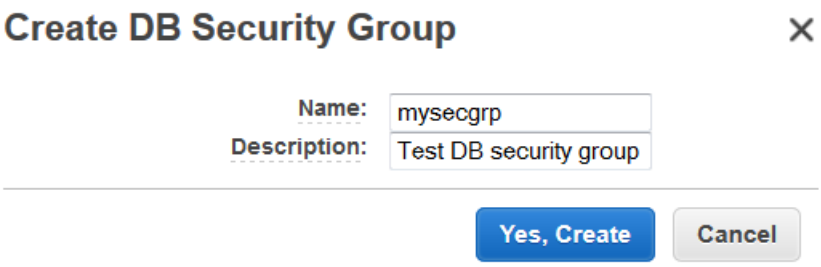
If **Supported Platforms** indicates *VPC*, your AWS account in the current region uses a default VPC. This means that you must create a VPC security group to enable access to a DB instance instead of a DB security group. For information on creating a VPC security group, see Security Groups for Your VPC.


Resources
You are using the following Amazon RDS resources in the US West (N. California) region:
DB Instances (0)
Reserved DB Purchases (0)
DB Snapshots (0)
Recent Events (8)
DB Parameter Groups (2)
Supported Platforms VPC
Default Network vpc-
Create Instance
Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud. You can click the button below to launch a Database (DB) Instance in...

3. Click **Security Groups** in the navigation pane on the left side of the window.
4. Click **Create DB Security Group**.



5. Type the name and description of the new DB security group in the **Name** and **Description** text boxes. Note that the security group name cannot contain spaces and cannot start with a number.



6. Click **Yes, Create**. The DB security group will be created. Note that a newly created DB security group does not provide access to a DB instance by default. You must specify a range of IP addresses or an Amazon EC2 security group that can have access to the DB instance. To specify IP addresses or an Amazon EC2 security group for a DB security group, see Authorizing Network Access to a DB Security Group from an IP Range (p. 543).

CLI

To create a DB security group

- Use the command `rds-create-db-security-group` with the following parameters:

```
PROMPT>rds-create-db-security-group mydbsecuritygroup -d "My new security
group"
```

Note that a newly created DB security group does not provide access to a DB instance by default. You must specify a range of IP addresses or an Amazon EC2 security group that can have access to the DB instance. To specify IP addresses or an Amazon EC2 security group for a DB security group, see Authorizing Network Access to a DB Security Group from an IP Range (p. 543).

API

To create a DB security group

- Call `CreateDBSecurityGroup` with the following parameters:
  - *`DBSecurityGroupName`* = `mydbsecuritygroup`
  - *`Description`* = `"My new security group"`

  Example

```
https://rds.amazonaws.com/
 ?Action=CreateDBSecurityGroup
 &DBParameterGroupName=mydbsecuritygroup
 &Description=My%20new%20db%20security%20group
 &Version=2012-01-15
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2012-01-20T22%3A06%3A23.624Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Note that a newly created DB security group does not provide access to a DB instance by default. You must specify a range of IP addresses or an Amazon EC2 security group that can have access to the DB instance. To specify IP addresses or an Amazon EC2 security group for a DB security group, see Authorizing Network Access to a DB Security Group from an IP Range (p. 543).

Listing Available DB Security Groups

You can list which DB security groups have been created for your AWS account.

AWS Management Console

To list all available DB security groups for an AWS account

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Click **Security Groups** in the navigation pane on the left side of the window.

   The available DB security groups appear in the **DB Security Groups** list.

CLI

To list all available DB security groups for an AWS account

- Use the command `rds-describe-db-security-groups` to list all available DB security groups for your AWS account.

```
PROMPT>rds-describe-db-security-groups
```

API

To list all available DB security groups for an AWS account

- Call `DescribeDBSecurityGroups` with no parameters.

  Example

```
https://rds.amazonaws.com/
 ?Action=DescribeDBSecurityGroups
 &MaxRecords=100
 &Version=2009-10-16
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

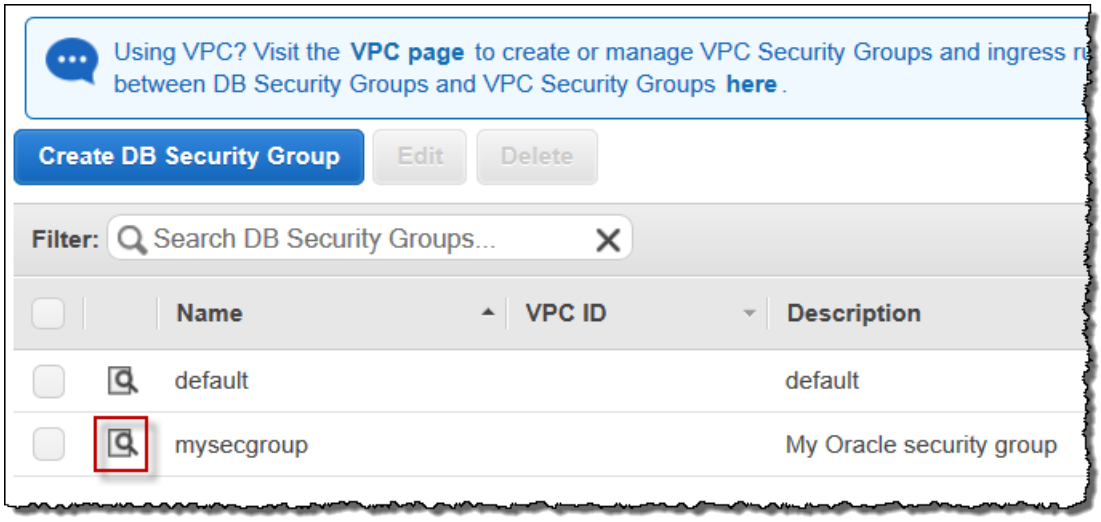
Viewing a DB security group

You can view detailed information about your DB security group to see what IP ranges have been authorized.

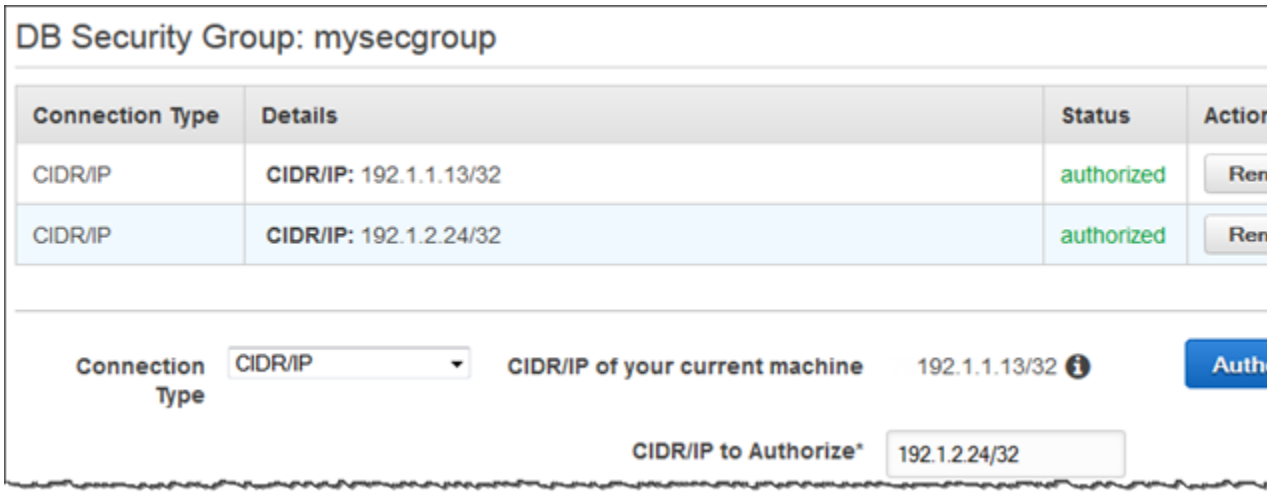
AWS Management Console

To view properties of a specific DB security group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Click **Security Groups** in the navigation pane on the left side of the window.
3. Select the details icon for the DB security group you want to view.



4. The detailed information for the DB security group is displayed.



CLI

To view properties of a specific DB security group

- Use the `rds-describe-db-security-groups` to view a DB security group. Specify the DB security group you want to view.

```
PROMPT>rds-describe-db-security-groups <mydbsecuritygroup>
```

API

To view properties of a specific DB security group

- Call `DescribeDBSecurityGroups` with the following parameters:
  - *`DBSecurityGroupName`* = `<mydbsecuritygroup>`

Example

```
https://rds.amazonaws.com/
 ?Action=DescribeDBSecurityGroups
 &DBParameterGroupName=mydbsecuritygroup
 &Version=2009-10-16
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-16T22%3A23%3A07.107Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Authorizing Network Access to a DB Security Group from an IP Range

By default, network access is turned off to a DB instance. If you want to access a DB instance that is not in a VPC, you must set access rules for a DB security group to allow access from specific EC2 security groups or CIDR IP ranges. You then must associate that DB instance with that DB security group. This process is called *ingress*. Once ingress is configured for a DB security group, the same ingress rules apply to all DB instances associated with that DB security group.

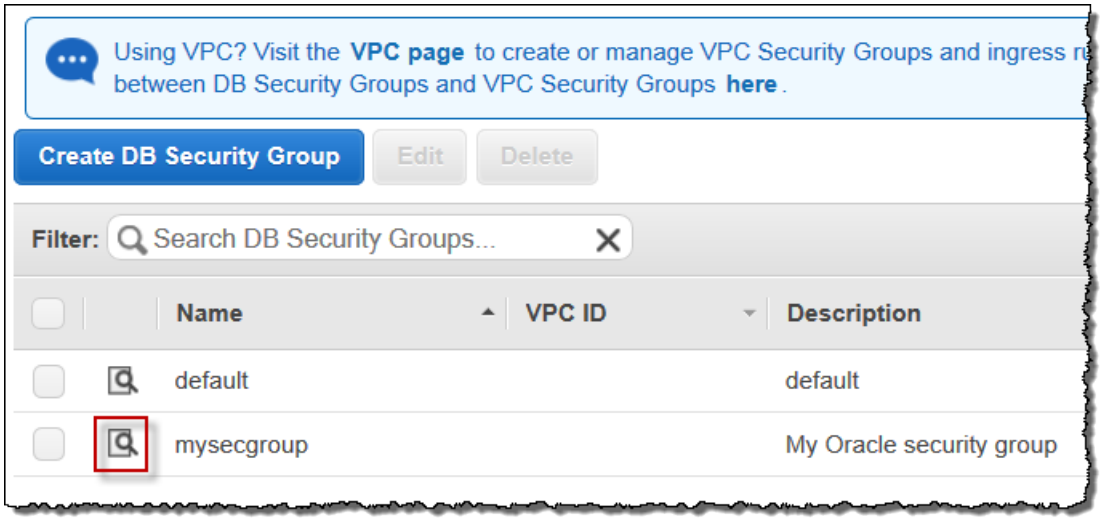
Caution
Talk with your network administrator if you are intending to access a DB instance behind a firewall to determine the IP addresses you should use.

In following example, you configure a DB security group with an ingress rule for a CIDR IP range.

AWS Management Console

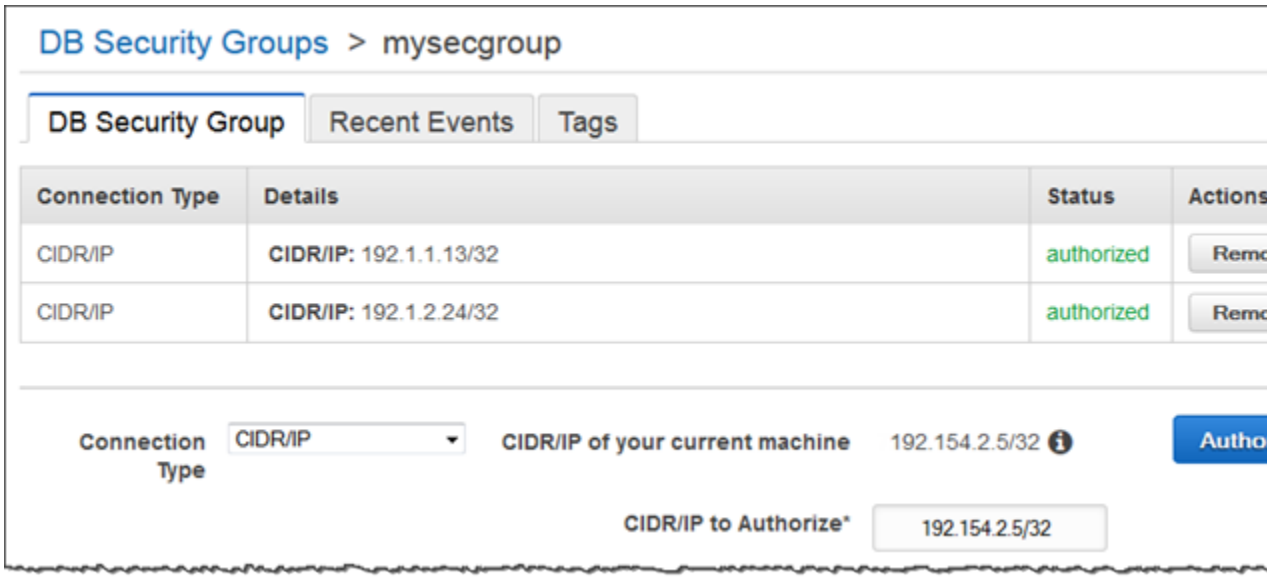
Configure a DB security group with an ingress rule for a CIDR IP range

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Select **Security Groups** from the navigation pane on the left side of the console window.
3. Select the details icon for the DB security group you want to authorize.



4. In the details page for your security group, select *CIDR/IP* from the **Connection Type** drop-down list, type the CIDR range for the ingress rule you would like to add to this DB security group into the **CIDR** text box, and click **Authorize**.

   Tip
   The AWS Management Console displays a CIDR IP based on your connection below the CIDR text field. If you are not accessing the DB instance from behind a firewall, this could be the CIDR IP you could use.



5. The status of the ingress rule will be **authorizing** until the new ingress rule has been applied to all DB instances that are associated with the DB security group that you modified. After the ingress rule has been successfully applied, the status will change to **authorized**.

CLI

To configure a DB security group with an ingress rule for a CIDR IP range

- Use the command `rds-authorize-db-security-group-ingress` to modify a DB security group.

```
PROMPT>rds-authorize-db-security-group-ingress mydbsecuritygroup --cidr-ip
 192.168.1.10/27
```

The command should produce output similar to the following:

```
SECGROUP  mydbsecuritygroup  My new DBSecurityGroup
IP-RANGE  192.168.1.10/27  authorizing
```

API

To configure a DB security group with an ingress rule for a CIDR IP range

- Call `AuthorizeDBSecurityGroupIngress` with the following parameters:
  - *`DBSecurityGroupName`* = `mydbsecuritygroup`
  - *`CIDRIP`* = `192.168.1.10/27`

Example

```
https://rds.amazonaws.com/
 ?CIDRIP=192.168.1.10%2F27
 &DBSecurityGroupName=mydbsecuritygroup
 &Version=2009-10-16
 &Action=AuthorizeDBSecurityGroupIngress
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-22T17%3A10%3A50.274Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Authorizing Network Access to a DB Instance from an Amazon EC2 Instance

If you want to access your DB instance from an Amazon EC2 instance, you must first determine if your EC2 instance and DB instance are in a VPC. If you are using a default VPC, you can assign the same EC2 or VPC security group that you used for your EC2 instance when you create or modify the DB instance that the EC2 instance will access.

If your DB instance and EC2 instance are not in a VPC, you must configure the DB instance's security group with an ingress rule that allows traffic from the Amazon EC2 instance. You would do this by adding the Amazon EC2 security group for the EC2 instance to the DB security group for the DB instance. In this example, you add an ingress rule to a DB security group for an Amazon EC2 security group.

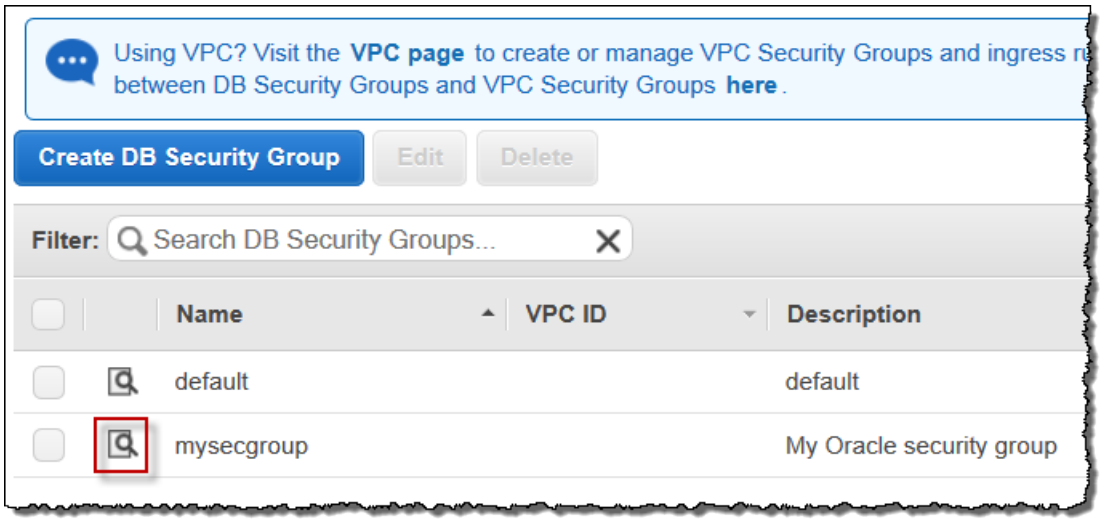
Important

- Adding an ingress rule to a DB security group for an Amazon EC2 security group only grants access to your DB instances from Amazon EC2 instances associated with that Amazon EC2 security group.
- You can't authorize an Amazon EC2 security group that is in a different AWS region than your DB instance. You can authorize an IP range, or specify an Amazon EC2 security group in the same region that refers to IP address in another region. If you specify an IP range, we recommend that you use the private IP address of your Amazon EC2 instance, which provides a more direct network route from your Amazon EC2 instance to your Amazon RDS DB instance, and does not incur network charges for data sent outside of the Amazon network.

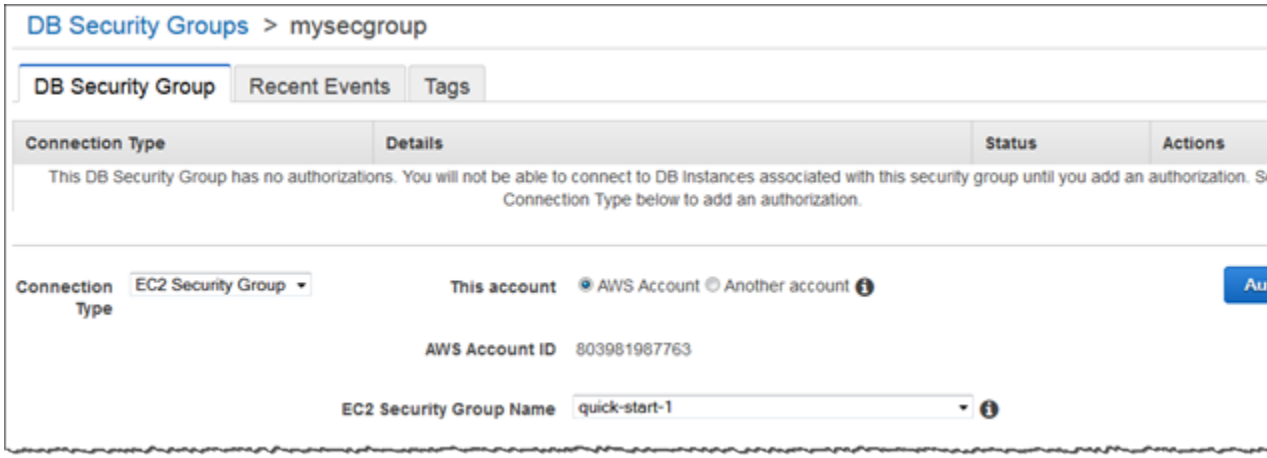
AWS Management Console

To add an EC2 security group to a DB security group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Select **Security Groups** from the navigation pane on the left side of the console window.
3. Select the details icon for the DB security group you want to grant access.



4. In the details page for your security group, select, select *EC2 Security Group* from the **Connection Type** drop-down list, and then select the Amazon EC2 security group you want to use. Then click **Authorize**.



5. The status of the ingress rule will be **authorizing** until the new ingress rule has been applied to all DB instances that are associated with the DB security group that you modified. After the ingress rule has been successfully applied, the status will change to **authorized**.

CLI

To grant access to an Amazon EC2 security group

- Use the command `rds-authorize-db-security-group-ingress` to grant access to an Amazon EC2 security group

```
PROMPT>rds-authorize-db-security-group-ingress default  --ec2-security-group-
name myec2group --ec2-security-group-owner-id 987654321021
```

The command should produce output similar to the following:

```
SECGROUP  Name     Description
SECGROUP  default  default
      EC2-SECGROUP  myec2group  987654321021  authorizing
```

API

To authorize network access to an Amazon EC2 security group

- Call `AuthorizeDBSecurityGroupIngress` with the following parameters:
  - *`EC2SecurityGroupName`* = `myec2group`
  - *`EC2SecurityGroupOwnerId`* = `987654321021`

Example

```
https://rds.amazonaws.com/
 ?Action=AuthorizeDBSecurityGroupIngress
 &EC2SecurityGroupOwnerId=987654321021
 &EC2SecurityGroupName=myec2group
 &Version=2009-10-16
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-22T17%3A10%3A50.274Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Revoking Network Access to a DB Instance from an IP Range

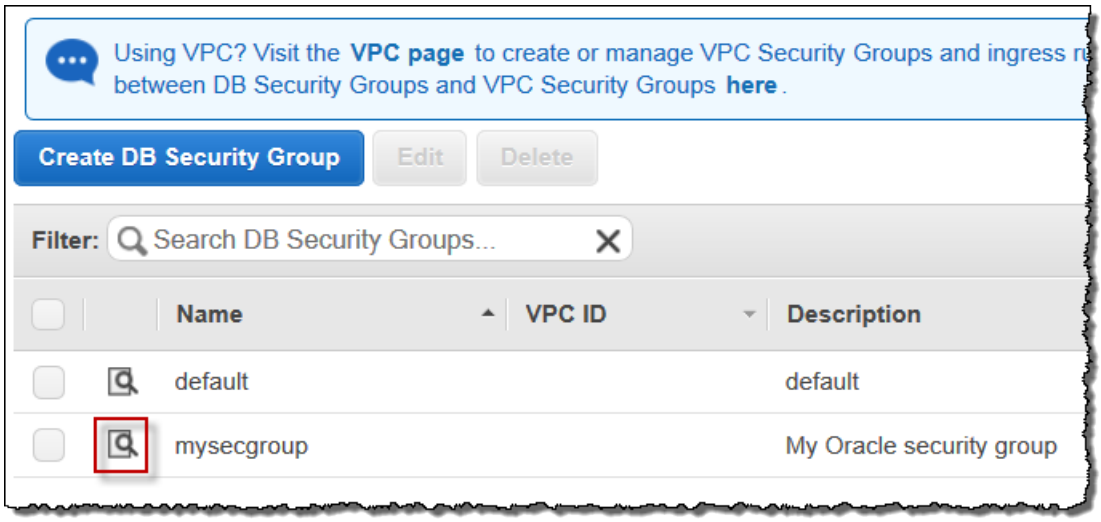
You can easily revoke network access from a CIDR IP range to DB Instances belonging to a DB security group by revoking the associated CIDR IP ingress rule.

In this example, you revoke an ingress rule for a CIDR IP on a DB Security Group.

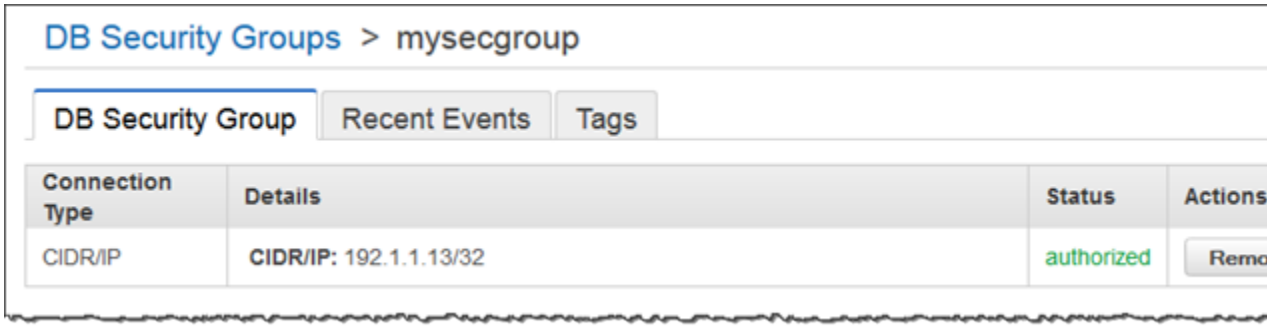
AWS Management Console

To revoke an ingress rule for a CIDR IP range on a DB Security Group.

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Select **Security Groups** from the navigation pane on the left side of the console window.
3. Select the details icon for the DB security group that has the ingress rule you want to revoke.



4. In the details page for your security group, select, click **Remove** next to the ingress rule you would like to revoke.



5. The status of the ingress rule will be **revoking** until the ingress rule has been removed from all DB instances that are associated with the DB security group that you modified. After the ingress rule has been successfully removed, the ingress rule will be removed from the DB security group.

CLI

To revoke an ingress rule for a CIDR IP range on a DB security group

- Use the command `rds-revoke-db-security-group-ingress` to modify a DB security group.

```
PROMPT>rds-revoke-db-security-group-ingress <mydbsecuritygroup> --cidr-ip
192.168.1.1/27
```

The command should produce output similar to the following:

```
SECGROUP  mydbsecuritygroup  My new DBSecurityGroup
  IP-RANGE  192.168.1.1/27  revoking
```

API

To revoke an ingress rule for a CIDR IP range on a DB security group

- Call `RevokeDBSecurityGroupIngress` with the following parameters:
  - *`DBSecurityGroupName`* = `<mydbsecuritygroup>`
  - *`CIDRIP`* = `192.168.1.10/27`

  Example

```
https://rds.amazonaws.com/
 ?Action=RevokeDBSecurityGroupIngress
 &DBSecurityGroupName=mydbsecuritygroup
 &CIDRIP=192.168.1.10%2F27
 &Version=2009-10-16
 &SignatureVersion=2&SignatureMethod=HmacSHA256
 &Timestamp=2009-10-22T22%3A32%3A12.515Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Related Topics

- Amazon RDS Security Groups (p. 110)

Working with Reserved DB Instances

Reserved DB instances let you reserve a DB instance for a one- or three-year term and in turn receive a significant discount on the hourly charge for instances that are covered by the reservation. You can use the command line tools, the API, or the AWS Management Console to list and purchase available reserved DB instance offerings.

Topics

- Getting Information About Available Reserved DB Instance Offerings (p. 551)
- Purchasing a Reserved DB Instance (p. 556)
- Getting Information About Your Account's Reserved DB Instances (p. 558)
- Related Topics (p. 561)

Reserved DB instances are available in three varieties—No Upfront, Partial Upfront, and All Upfront—that let you optimize your Amazon RDS costs based on your expected usage. For more information about reserved DB instance types, see Amazon RDS Reserved Instances.

No Upfront
: This option provides access to a reserved DB instance without requiring an upfront payment. Your No Upfront reserved DB instance will bill a discounted hourly rate for every hour within the term, regardless of usage, and no upfront payment is required. This option is only available as a one-year reservation.

Partial Upfront
: This option requires a part of the reserved DB instance to be paid upfront. The remaining hours in the term are billed at a discounted hourly rate, regardless of usage. This option is the replacement for the previous Heavy Utilization option.

All Upfront
: Full payment is made at the start of the term, with no other costs incurred for the remainder of the term regardless of the number of hours used.

Remember that discounted usage fees for reserved DB instance purchases are tied to instance type and region. Also, you can move reserved DB instances from an EC2-Classic (non-VPC) instance into an Amazon Virtual Private Cloud (Amazon VPC) without additional charge.

If you shut down a running DB instance on which you have been getting a discounted rate as a result of a reserved DB instance purchase, and the term of the reserved DB instance has not yet expired, you will continue to get the discounted rate if you launch another DB instance with the same specifications during the term. Your upfront payment for a reserved DB instance will reserve the resources for your use. Because these resources are reserved for you, you will be billed for the resources regardless of whether you use them.

The Medium Utilization and Light Utilization reserved DB instance options are still available for Amazon RDS, but will be discontinued on August 15, 2015. The following table summarizes these offerings.

Medium Utilization
: These reserved DB instances are the best option if you plan to use your reserved DB instances a substantial amount of the time, but want either a lower one-time fee or the flexibility to stop paying for your DB instance when you shut it off. This offering type is equivalent to the reserved DB instance offering available before the 2011-12-19 API version of Amazon RDS. Medium Utilization reserved DB instances are a more cost-effective option when you plan to run more than 40 percent of the reserved DB instance term. This option can save you up to 49 percent off of the On-Demand price. With Medium Utilization reserved DB instances, you pay a slightly higher one-time fee than with Light Utilization reserved DB instances, and you receive lower hourly usage rates when you run a DB instance.

Light Utilization

These reserved DB instances are ideal for periodic workloads that run only a couple of hours a day or a few days per week. Using Light Utilization reserved DB instances, you pay a one-time fee followed by a discounted hourly usage fee when your DB instance is running. You can start saving when your instance is running more than 17 percent of the reserved DB instance term, and you can save up to 33 percent off of the On-Demand rates over the entire term of your reserved DB instance.

Getting Information About Available Reserved DB Instance Offerings

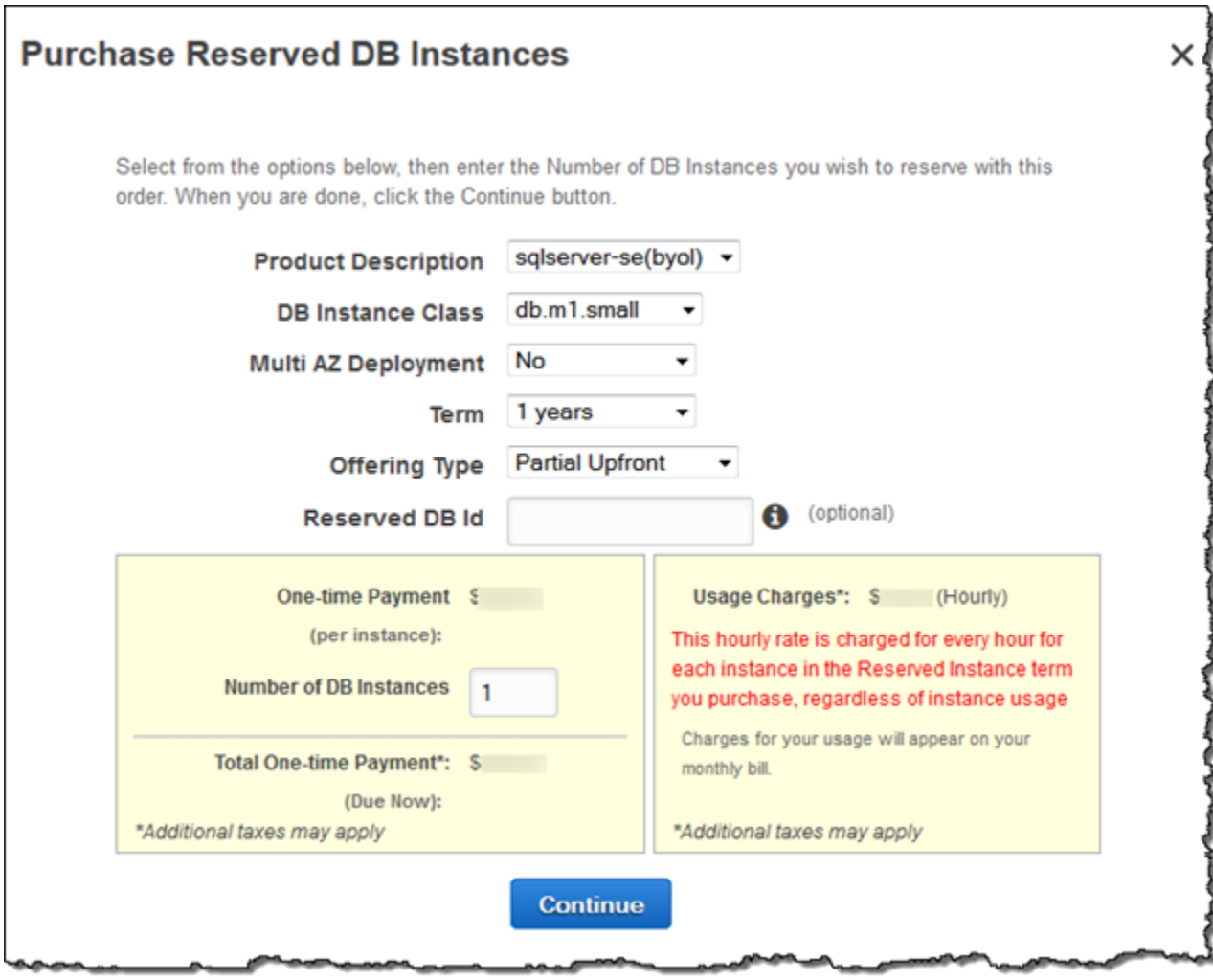
Before you purchase a reserved DB instance, you can get pricing and information about available reserved DB instance offerings.

The following example shows how to do so.

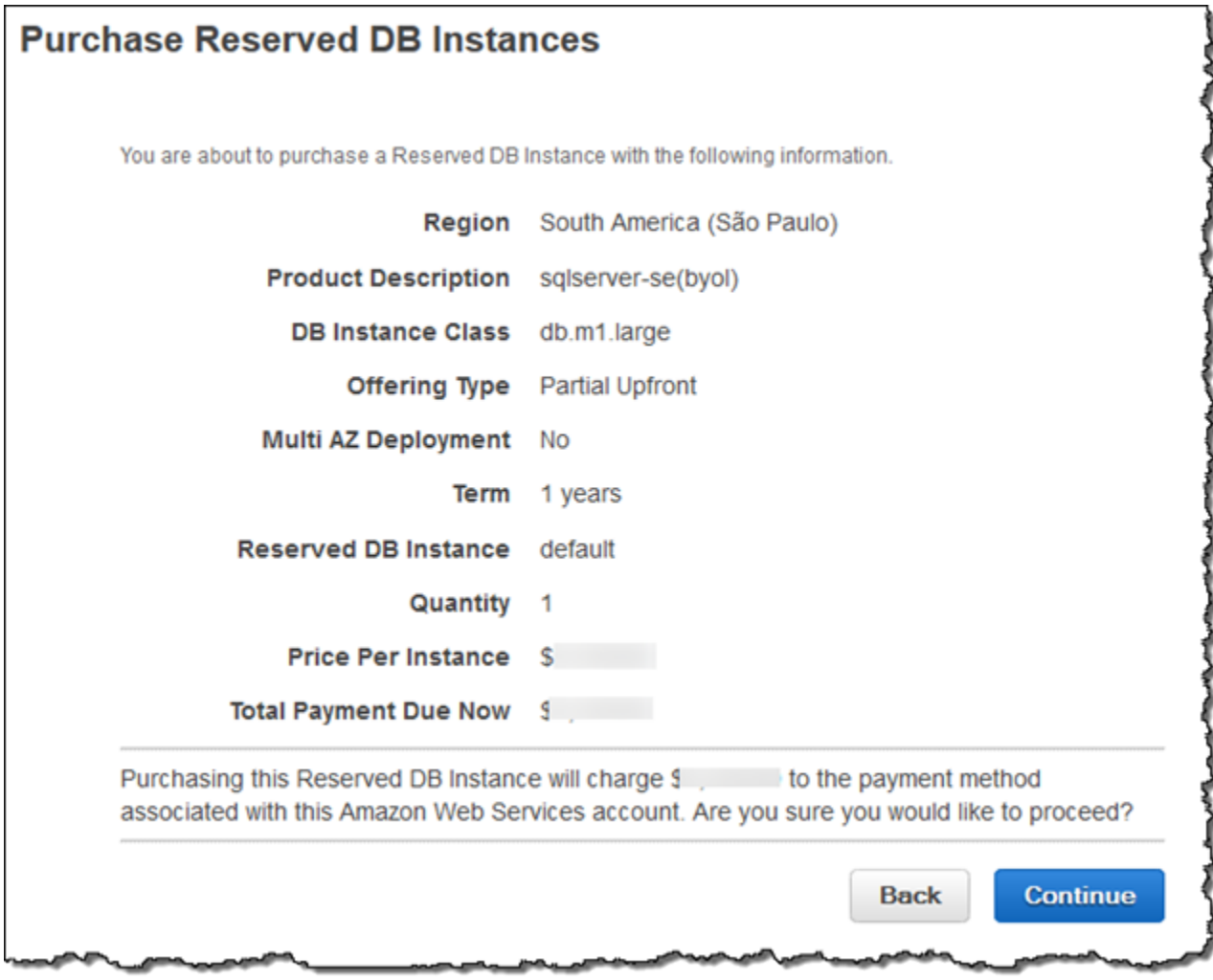
AWS Management Console

To get pricing and information about available reserved DB instances

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click the **Reserved DB Purchases** link.
3. Choose **Purchase Reserved DB Instance**.
4. For **Product Description**, choose the DB engine and licensing type.
5. For **DB Instance Class**, choose the DB instance class.
6. For **Multi-AZ Deployment**, choose whether or not you want a Multi-AZ deployment.
7. For **Term**, choose the length of time you want the DB instance reserved.
8. For **Offering Type**, choose the offering type.
9. Information is displayed after you select the offering type. When you have selected the reserved DB instance you want, choose **Continue**.


Purchase Reserved DB Instances
Select from the options below, then enter the Number of DB Instances you wish to reserve with this order. When you are done, click the Continue button.
Product Description
sqlserver-se(byol)
DB Instance Class
db.m1.small
Multi AZ Deployment
No
Term
1 years
Offering Type
Partial Upfront
Reserved DB Id
(optional)
One-time Payment (per instance):
Number of DB Instances
1
Total One-time Payment*: (Due Now):
*Additional taxes may apply
Usage Charges*: (Hourly)
This hourly rate is charged for every hour for each instance in the Reserved Instance term you purchase, regardless of instance usage
Charges for your usage will appear on your monthly bill.
*Additional taxes may apply
Continue

10. The summation screen shows you the instance information and cost. Click the **X** in the upper-right corner of the page to avoid incurring any charges.



CLI

To get information about reserved DB instances

- Type the following command at a command prompt:

```
PROMPT>rds-describe-reserved-db-instances-offerings  --headers
```

This call returns output similar to the following:

```
OFFERING  OfferingId                            Class         Multi-AZ
Duration  Fixed Price  Usage Price  Description  Offering Type
OFFERING  438012d3-4052-4cc7-b2e3-8d3372e0e706  db.m1.large   y         1y
        1820.00 USD  0.368 USD    mysql        Partial Upfront
OFFERING  649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  db.m1.small   n         1y
```

```
         227.50 USD  0.046 USD    mysql        Partial Upfront
OFFERING  123456cd-ab1c-47a0-bfa6-12345667232f  db.m1.small   n         1y
         162.00 USD   0.00 USD    mysql        All Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   0.123   USD       Hourly
OFFERING  123456cd-ab1c-37a0-bfa6-12345667232d  db.m1.large   y         1y
         700.00 USD   0.00 USD    mysql        All Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   1.25    USD       Hourly
OFFERING  123456cd-ab1c-17d0-bfa6-12345667234e  db.m1.xlarge  n         1y
        4242.00 USD   2.42 USD    mysql        No Upfront
```

API

To get information about available reserved DB Instances

- Call `DescribeReservedDBInstancesOfferings`.

  Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=DescribeReservedDBInstancesOfferings
   &ReservedDBInstancesOfferingId=438012d3-4052-4cc7-b2e3-8d3372e0e706
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140411/us-east-1/rds/aws4_request
   &X-Amz-Date=20140411T203327Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signature=545f04acf
feb4b80d2e778526b1c9da79d0b3097151c24f28e83e851d65422e2
```

  This call returns output similar to the following:

```
<DescribeReservedDBInstancesOfferingsResponse xmlns="http://rds.amazon
aws.com/doc/2014-09-01/">
  <DescribeReservedDBInstancesOfferingsResult>
    <ReservedDBInstancesOfferings>
      <ReservedDBInstancesOffering>
        <Duration>31536000</Duration>
        <OfferingType>Partial Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <FixedPrice>1820.0</FixedPrice>
        <ProductDescription>mysql</ProductDescription>
        <UsagePrice>0.368</UsagePrice>
        <MultiAZ>true</MultiAZ>
        <ReservedDBInstancesOfferingId>438012d3-4052-4cc7-b2e3-
8d3372e0e706</ReservedDBInstancesOfferingId>
        <DBInstanceClass>db.m1.large</DBInstanceClass>
```

```
      </ReservedDBInstancesOffering>
      <ReservedDBInstancesOffering>
        <Duration>31536000</Duration>
        <OfferingType>Partial Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <FixedPrice>227.5</FixedPrice>
        <ProductDescription>mysql</ProductDescription>
        <UsagePrice>0.046</UsagePrice>
        <MultiAZ>false</MultiAZ>
        <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-
060f8e75e95f</ReservedDBInstancesOfferingId>
        <DBInstanceClass>db.m1.small</DBInstanceClass>
      </ReservedDBInstancesOffering>
      <ReservedDBInstancesOffering>
        <Duration>31536000</Duration>
        <OfferingType>All Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges>
          <RecurringCharge>
            <RecurringChargeFrequency>Hourly</RecurringChargeFrequency>
            <RecurringChargeAmount>0.123</RecurringChargeAmount>
          </RecurringCharge>
        </RecurringCharges>
        <FixedPrice>162.0</FixedPrice>
        <ProductDescription>mysql</ProductDescription>
        <UsagePrice>0.0</UsagePrice>
        <MultiAZ>false</MultiAZ>
       <ReservedDBInstancesOfferingId>TEMP-DELETE-1</ReservedDBInstancesOf
feringId>
        <DBInstanceClass>db.m1.small</DBInstanceClass>
      </ReservedDBInstancesOffering>
      <ReservedDBInstancesOffering>
        <Duration>31536000</Duration>
        <OfferingType>All Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges>
          <RecurringCharge>
            <RecurringChargeFrequency>Hourly</RecurringChargeFrequency>
            <RecurringChargeAmount>1.25</RecurringChargeAmount>
          </RecurringCharge>
        </RecurringCharges>
        <FixedPrice>700.0</FixedPrice>
        <ProductDescription>mysql</ProductDescription>
        <UsagePrice>0.0</UsagePrice>
        <MultiAZ>true</MultiAZ>
       <ReservedDBInstancesOfferingId>TEMP-DELETE-2</ReservedDBInstancesOf
feringId>
        <DBInstanceClass>db.m1.large</DBInstanceClass>
      </ReservedDBInstancesOffering>
      <ReservedDBInstancesOffering>
        <Duration>31536000</Duration>
        <OfferingType>No Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <FixedPrice>4242.0</FixedPrice>
        <ProductDescription>mysql</ProductDescription>
        <UsagePrice>2.42</UsagePrice>
```

```
        <MultiAZ>false</MultiAZ>
       <ReservedDBInstancesOfferingId>TEMP-DELETE-3</ReservedDBInstancesOf
feringId>
        <DBInstanceClass>db.m1.xlarge</DBInstanceClass>
      </ReservedDBInstancesOffering>
    </ReservedDBInstancesOfferings>
  </DescribeReservedDBInstancesOfferingsResult>
  <ResponseMetadata>
    <RequestId>5e4ec40b-2978-11e1-9e6d-771388d6ed6b</RequestId>
  </ResponseMetadata>
</DescribeReservedDBInstancesOfferingsResponse>
```

Purchasing a Reserved DB Instance

The following example shows how to purchase a reserved DB instance offering.

Important
Following the examples in this section will incur charges on your AWS account.

AWS Management Console

The following example shows purchasing a specific reserved DB instance offering, *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*, with a reserved DB instance ID of *myreservationID*.

To purchase a reserved DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the **Navigation** pane, click the **Reserved DB Instances** link.
3. Choose **Purchase Reserved DB Instance**.
4. For **Product Description**, choose the DB engine type.
5. For **DB Instance Class**, choose the DB instance class.
6. For **Multi-AZ Deployment**, choose whether or not you want a Multi-AZ deployment.
7. For **Term**, choose the length of time you want the DB instance reserved.
8. For **Offering Type**, choose the offering type.
9. (Optional.) To purchase or repurchase a specific reserved DB instance, type the specific reserved DB instance ID for **Reserved DB ID**.
10. Choose **Continue**.

    The **Purchase Reserved DB Instance** dialog box appears, showing a summary of the reserved DB instance attributes that you've selected and the payment due.
11. To proceed and purchase the reserved DB instance, choose **Yes, Purchase**.

CLI

This example shows purchasing a specific reserved DB instance offering, *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*, with a reserved DB instance ID of *myreservationID*.

To purchase a reserved DB instance

- Type the following command at a command prompt:

```
PROMPT>rds-purchase-reserved-db-instances-offering 649fd0c8-cf6d-47a0-bfa6-
060f8e75e95f -i myreservationID
```

The command returns output similar to the following:

```
RESERVATION  ReservationId      Class        Multi-AZ  Start Time
      Duration  Fixed Price  Usage Price  Count  State          Description
  Offering Type
RESERVATION  myreservationid    db.m1.small  y         2011-12-
19T00:30:23.247Z  1y        455.00 USD   0.092 USD    1      payment-pending
  mysql        Partial Upfront
```

API

The following example shows purchasing a specific reserved DB instance offering, *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*, with a reserved DB instance ID of *myreservationID*.

To purchase a reserved DB instance

- Call `PurchaseReservedDBInstancesOffering` with the following parameters:
  - *`ReservedDBInstancesOfferingId`* = `649fd0c8-cf6d-47a0-bfa6-060f8e75e95f`
  - *`ReservedDBInstanceID`* = `myreservationID`
  - *`DBInstanceCount`* = `1`

Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=PurchaseReservedDBInstancesOffering
   &ReservedDBInstanceId=myreservationID
   &ReservedDBInstancesOfferingId=438012d3-4052-4cc7-b2e3-8d3372e0e706
   &DBInstanceCount=10
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140415/us-east-1/rds/aws4_request
   &X-Amz-Date=20140415T232655Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=c2ac761e8c8f54a8c0727f5a87ad0a766fbb0024510b9aa34ea6d1f7df52fb11
```

This call returns output similar to the following:

```
<PurchaseReservedDBInstancesOfferingResponse xmlns="http://rds.amazon
aws.com/doc/2014-09-01/">
  <PurchaseReservedDBInstancesOfferingResult>
    <ReservedDBInstance>
      <OfferingType>Partial Upfront</OfferingType>
      <CurrencyCode>USD</CurrencyCode>
      <RecurringCharges/>
      <ProductDescription>mysql</ProductDescription>
      <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-
060f8e75e95f</ReservedDBInstancesOfferingId>
      <MultiAZ>true</MultiAZ>
      <State>payment-pending</State>
      <ReservedDBInstanceId>myreservationID</ReservedDBInstanceId>
      <DBInstanceCount>10</DBInstanceCount>
      <StartTime>2011-12-18T23:24:56.577Z</StartTime>
      <Duration>31536000</Duration>
      <FixedPrice>123.0</FixedPrice>
      <UsagePrice>0.123</UsagePrice>
      <DBInstanceClass>db.m1.small</DBInstanceClass>
    </ReservedDBInstance>
  </PurchaseReservedDBInstancesOfferingResult>
  <ResponseMetadata>
    <RequestId>7f099901-29cf-11e1-bd06-6fe008f046c3</RequestId>
  </ResponseMetadata>
</PurchaseReservedDBInstancesOfferingResponse>
```

Getting Information About Your Account's Reserved DB Instances

You can get information about reserved DB instances for your AWS account as described following.

AWS Management Console

To get information about reserved DB instances for your AWS account

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the **Navigation** pane, click the **Reserved DB Instances** link.

   The reserved DB instances for your account appear in the **My Reserved DB Instances** list. You can choose any of the reserved DB instances in the list to see detailed information about that reserved DB instance in the detail pane at the bottom of the console.

CLI

To get information about reserved DB instances for your AWS account

- Type the following command at a command prompt:

```
PROMPT>rds-describe-reserved-db-instances --headers
```

This command should return output similar to the following:

```
RESERVATION  ReservationId     Class        Multi-AZ  Start Time
      Duration  Fixed Price  Usage Price  Count  State    Description  Of
fering Type
RESERVATION  ki-real-ri-test5  db.m1.small  y        2011-12-09T23:37:44.720Z
  1y        455.00 USD   0.092 USD    1      retired  mysql        Partial
 Upfront
```

API

To get information about reserved DB instances for your AWS account

- Call `DescribeReservedDBInstances`.

Example

```
https://rds.us-west-2.amazonaws.com/
   ?Action=DescribeReservedDBInstances
   &ReservedDBInstanceId=customerSpecifiedID
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140420/us-west-2/rds/aws4_request
   &X-Amz-Date=20140420T162211Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=3312d17a4c43bcd209bc22a0778dd23e73f8434254abbd7ac53b89ade3dae88e
```

The API returns output similar to the following:

```
<DescribeReservedDBInstancesResponse xmlns="http://rds.amazonaws.com/doc/2014-
09-01/">
  <DescribeReservedDBInstancesResult>
    <ReservedDBInstances>
      <ReservedDBInstance>
        <OfferingType>Partial Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <ProductDescription>mysql</ProductDescription>
        <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-
060f8e75e95f</ReservedDBInstancesOfferingId>
        <MultiAZ>false</MultiAZ>
        <State>payment-failed</State>
        <ReservedDBInstanceId>myreservationid</ReservedDBInstanceId>
        <DBInstanceCount>1</DBInstanceCount>
        <StartTime>2010-12-15T00:25:14.131Z</StartTime>
        <Duration>31536000</Duration>
        <FixedPrice>227.5</FixedPrice>
        <UsagePrice>0.046</UsagePrice>
        <DBInstanceClass>db.m1.small</DBInstanceClass>
      </ReservedDBInstance>
      <ReservedDBInstance>
        <OfferingType>Partial Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <ProductDescription>mysql</ProductDescription>
        <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-
060f8e75e95f</ReservedDBInstancesOfferingId>
        <MultiAZ>false</MultiAZ>
        <State>payment-failed</State>
        <ReservedDBInstanceId>myreservationid2</ReservedDBInstanceId>
        <DBInstanceCount>1</DBInstanceCount>
        <StartTime>2010-12-15T01:07:22.275Z</StartTime>
        <Duration>31536000</Duration>
        <FixedPrice>227.5</FixedPrice>
        <UsagePrice>0.046</UsagePrice>
```

```
        <DBInstanceClass>db.m1.small</DBInstanceClass>
      </ReservedDBInstance>
    </ReservedDBInstances>
  </DescribeReservedDBInstancesResult>
  <ResponseMetadata>
    <RequestId>23400d50-2978-11e1-9e6d-771388d6ed6b</RequestId>
  </ResponseMetadata>
</DescribeReservedDBInstancesResponse>
```

Related Topics

- How You Are Charged for Amazon RDS (p. 5)

Using Amazon RDS with Amazon Virtual Private Cloud (VPC)

A virtual private cloud is a virtual network that is logically isolated from other virtual networks in the AWS cloud. Amazon Virtual Private Cloud (VPC) lets you launch AWS resources, such as an Amazon RDS or Amazon EC2 instance, into a VPC. The VPC can either be a default VPC that comes with your account or it could be one that you create. All VPCs are associated with your AWS account.

Amazon RDS supported two EC2 platforms: *EC2-VPC* and *EC2-Classic*. The *EC2-VPC* platform has a default VPC where all new DB instances are created unless you specify otherwise. The EC2-Classic platform does not have a default VPC, but as with either platform, you can create your own VPC and specify that a DB instance be located in that VPC. If you are a new customer to Amazon RDS or if you are using a region you have not previously used, you are most likely working with the *EC2-VPC* platform.

When an option group is assigned to a DB instance, it is linked to the supported platform the DB instance is on, either VPC or EC2-Classic (non-VPC). Furthermore, if a DB instance is in a VPC, the option group associated with the instance is linked to that VPC. This means that you cannot use the option group assigned to a DB instance if you attempt to restore the instance into a different VPC or onto a different platform.

If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance. Note that with persistent or permanent options, such as Oracle TDE, you must create a new option group that includes the persistent or permanent option when restoring a DB instance into a different VPC.

Amazon Virtual Private Cloud (VPC) is an AWS service and this section only covers VPC topics that directly affect access to a DB instance. You should read the Amazon VPC documentation to familiarize yourself with all the features of a VPC. For more information about Amazon VPC, see the table below that provides several links to the Amazon VPC documentation.

The following topics in this section apply to working with DB instances inside a VPC:

Topics

- Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 563)
- Working with a DB Instance in a VPC (p. 564)
- Working with DB Subnet Groups (p. 565)
- Hiding a DB Instance in a VPC from the Internet (p. 565)
- Creating a DB Instance in a VPC (p. 566)
- Moving a DB Instance not in a VPC into a VPC (p. 568)
- Using ClassicLink to Connect an EC2-Classic Instance to a DB Instance in an Amazon VPC (p. 569)

Amazon VPC has its own set of documentation that describes how to create and use a VPC. The following table shows links to the Amazon VPC guides.

Description	Documentation
How to get started using Amazon VPC	Amazon VPC Getting Started Guide
How to use Amazon VPC through the AWS Management Console	Amazon VPC User Guide

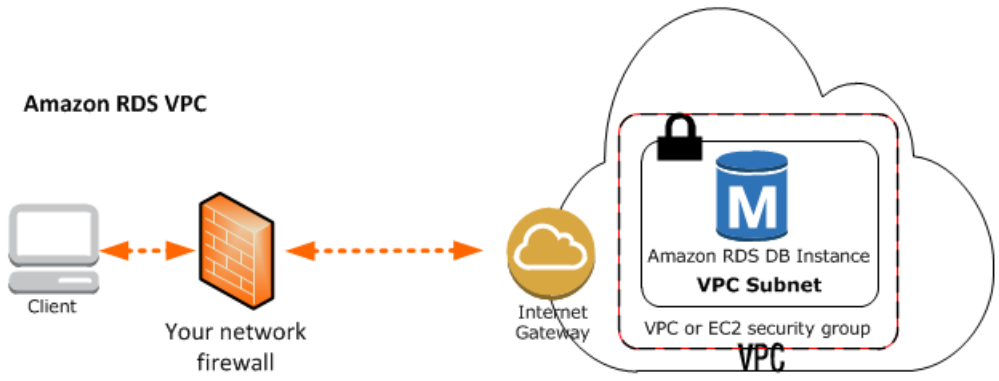
Description	Documentation
Complete descriptions of all the Amazon VPC commands	Amazon EC2 Command Line Reference (the Amazon VPC commands are part of the Amazon EC2 reference)
Complete descriptions of the Amazon VPC API actions, data types, and errors	Amazon EC2 API Reference (the Amazon VPC API actions are part of the Amazon EC2 reference)
Information for the network administrator who needs to configure the gateway at your end of an optional IPsec VPN connection	Amazon VPC Network Administrator Guide

Determining Whether You are Using the EC2-VPC or EC2-Classic Platform

You can create a DB instance on two different platforms: *EC2-Classic* or *EC2-VPC*. The type of platform determines if you have a default VPC and which type of security group you use to provide access to your DB instance. The *EC2-Classic* platform is the original platform used by Amazon RDS; if you want to use a VPC with this platform, you must create it using Amazon VPC. The *EC2-VPC* platform provides a default VPC (virtual private cloud) where a new DB instance is created, and you must use either an EC2 or VPC security group to provide access to the DB instance.

> **Note**
> If you are a new Amazon RDS customer, if you have never created a DB instance before, or if you are creating a DB instance in a region you have not used before, in almost all cases you are on the *EC2-VPC* platform and have a default VPC.

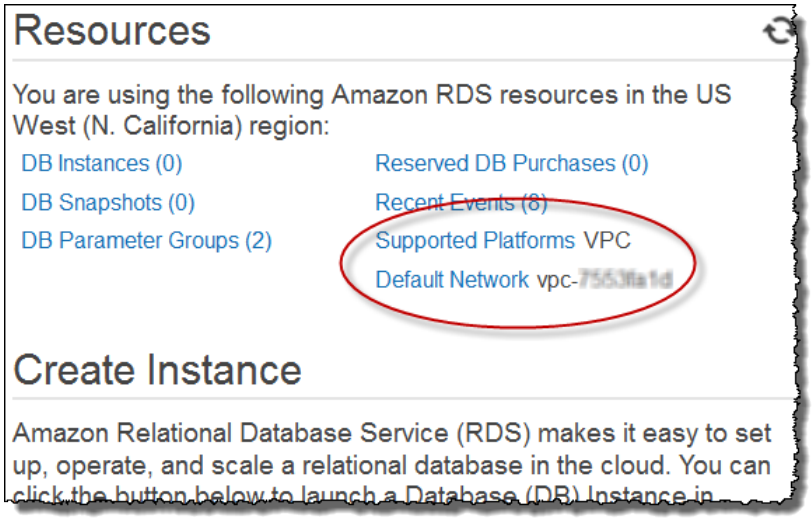
The following illustration shows how you access a DB instance when it is on the *EC2-VPC* platform and in a default VPC:



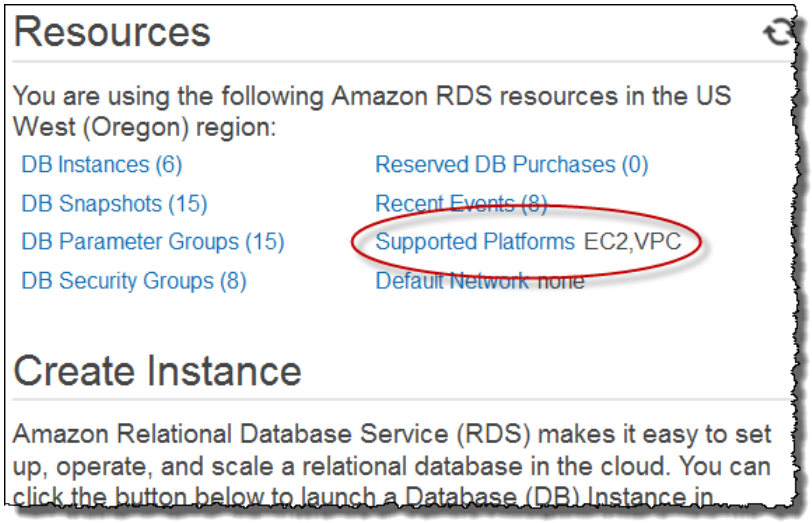
You can tell which platform your AWS account in a given region is using by looking at the RDS console or EC2 console home pages. If you are a new Amazon RDS customer, if you have never created a DB instance before, or if you are creating a DB instance in a region you have not used before, you may be redirected to the Getting Started page and will not see the page below.

If **Supported Platforms** indicates *VPC*, your AWS account in the current region uses the *EC2-VPC* platform, and uses a default VPC. The name of the default VPC is shown below the supported platform.

To provide access to a DB instance created on the *EC2-VPC* platform, you must create an EC2 or VPC security group.


Resources
You are using the following Amazon RDS resources in the US West (N. California) region:
DB Instances (0)
Reserved DB Purchases (0)
DB Snapshots (0)
Recent Events (8)
DB Parameter Groups (2)
Supported Platforms VPC
Default Network vpc-
Create Instance
Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud. You can click the button below to launch a Database (DB) Instance in

If **Supported Platforms** indicates *EC2, VPC*, your AWS account in the current region uses the *EC2-Classic* platform, and you do not have a default VPC. To provide access to a DB instance created on the *EC2-Classic* platform, you must create a DB security group. Note that you can create a VPC on the *EC2-Classic* platform, but one is not created for you by default as it is on the *EC2-VPC* platform.


Resources
You are using the following Amazon RDS resources in the US West (Oregon) region:
DB Instances (6)
Reserved DB Purchases (0)
DB Snapshots (15)
Recent Events (8)
DB Parameter Groups (15)
Supported Platforms EC2,VPC
DB Security Groups (8)
Default Network none
Create Instance
Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud. You can click the button below to launch a Database (DB) Instance in

Working with a DB Instance in a VPC

Here are some tips on working with a DB instance in a VPC:

- Your VPC must have at least one subnet in at least two of the Availability Zones in the region where you want to deploy your DB instance. A subnet is a segment of a VPC's IP address range that you can specify and that lets you group instances based on your security and operational needs.
- If you want your DB instance in the VPC to be publicly accessible, you must enable the VPC attributes *DNS hostnames* and *DNS resolution*.
- Your VPC must have a DB subnet group that you create (for more information, see the next section). You create a DB subnet group by specifying the subnets you created. Amazon RDS uses that DB

subnet group and your preferred Availability Zone to select a subnet and an IP address within that subnet to assign to your DB instance.

- Your VPC must have a VPC security group that allows access to the DB instance.
- The CIDR blocks in each of your subnets must be large enough to accommodate spare IP addresses for Amazon RDS to use during maintenance activities, including failover and compute scaling.
- A VPC can have an *instance tenancy* attribute of either *default* or *dedicated*. All default VPCs have the instance tenancy attribute set to default, and a default VPC can support any DB instance class.

  If you choose to have your DB instance in a dedicated VPC where the instance tenancy attribute is set to dedicated, the DB instance class of your DB instance must be one of the approved Amazon EC2 dedicated instance types. For example, the m3.medium EC2 dedicated instance corresponds to the db.m3.medium DB instance class. For example, For more information about the instance types that can be in a dedicated instance, see Amazon EC2 Dedicated Instances. For information about instance tenancy in a VPC, see Using EC2 Dedicated Instances in the *Amazon Virtual Private Cloud User Guide*.

The most common scenarios for using a VPC are documented at Scenarios for Using Amazon VPC. Each of these scenarios have a link to a detailed explanation of the scenario. At the end of the section is a section called **Implementing the Scenario** that gives you instructions on how to create a VPC for that scenario. For detailed instructions on creating a VPC, see the Amazon VPC User Guide.

Working with DB Subnet Groups

Subnets are segments of a VPC's IP address range that you designate to group your resources based on security and operational needs. A DB subnet group is a collection of subnets (typically private) that you create in a VPC and that you then designate for your DB instances. A DB subnet group allows you to specify a particular VPC when creating DB instances using the CLI or API; if you use the console, you can just select the VPC and subnets you want to use.

Each DB subnet group should have subnets in at least two Availability Zones in a given region. If you are using SQL Server Mirroring with a SQL Server DB instance in a VPC, you must create a DB subnet group that has 3 subnets in distinct Availability Zones. When creating a DB instance in VPC, you must select a DB subnet group. Amazon RDS uses that DB subnet group and your preferred Availability Zone to select a subnet and an IP address within that subnet to associate with your DB instance. If the primary DB instance of a Multi-AZ deployment fails, Amazon RDS can promote the corresponding standby and subsequently create a new standby using an IP address of the subnet in one of the other Availability Zones.

When Amazon RDS creates a DB instance in a VPC, it assigns a network interface to your DB instance by using an IP address selected from your DB Subnet Group. However, we strongly recommend that you use the DNS Name to connect to your DB instance because the underlying IP address can change during failover.

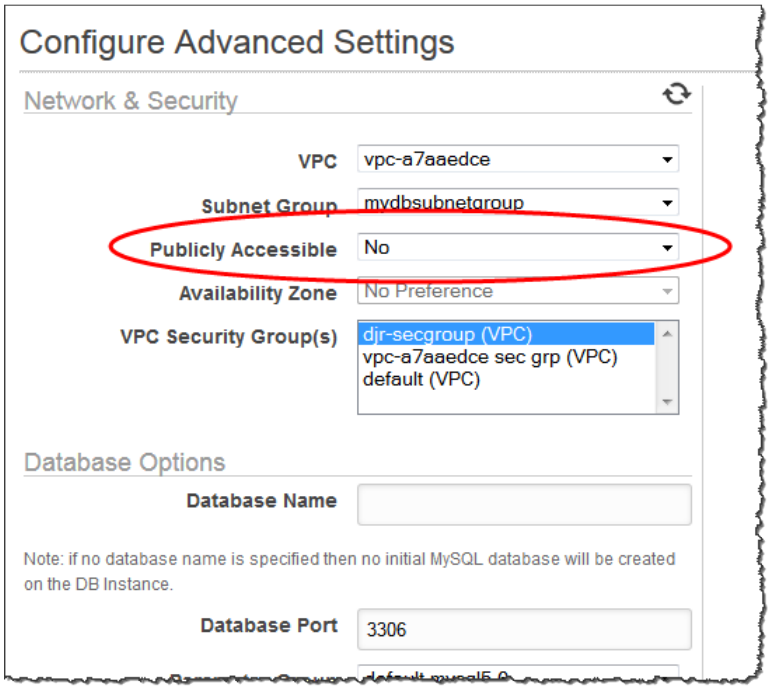
Note
For each DB instance that you run in a VPC, you should reserve at least one address in each subnet in the DB subnet group for use by Amazon RDS for recovery actions.

Hiding a DB Instance in a VPC from the Internet

One common Amazon RDS scenario is to have a VPC in which you have an EC2 instance with a public-facing web application and a DB instance with a database that is not publicly accessible. For example, you could create a VPC that has a public subnet and a private subnet. Amazon EC2 instances that function as web servers could be deployed in the public subnet, and the Amazon RDS DB instances would be deployed in the private subnet. In such a deployment, only the web servers have access to the DB instances.

When you launch a DB instance inside any VPC (including a default VPC), you can designate whether the DB instance you create has a DNS that resolves to a public IP address by using the *PubliclyAccessible* parameter. This parameter lets you designate whether there is public access to the DB instance, even if the VPC has a public IP address. Note that access to the DB instance is ultimately controlled by the security group it uses, and that public access is not permitted if the security group assigned to the DB instance does not permit it. If you want a DB instance in a VPC to be publicly accessible, you must also enable the VPC attributes *DNS hostnames* and *DNS resolution*. For more information about creating a VPC, see Working with a DB Instance in a VPC (p. 564).

The following illustration shows the **Publicly Accessible** parameter in the **Launch DB Instance Wizard**.



Creating a DB Instance in a VPC

DB instances in a VPC can require slightly more set up than a DB instance not in a VPC, but the added flexibility is often worth it. If your account has a default VPC, you can begin with step 3 in this tutorial since the VPC and DB subnet group have been created for you. If your AWS account does not have a default VPC or if you do not have a default VPC in a particular region, you can create a VPC using the Amazon VPC service and launch a DB instance in the VPC. If you don't know if you have a default VPC, see the topic Determining Whether You are Using the EC2-VPC or EC2-Classic Platform (p. 563)

You should have a clear understanding of the type of scenario you intend for your VPC. The most common scenarios for using a VPC are documented at Scenarios for Using Amazon VPC. Each of these scenarios have a link to a detailed explanation of the scenario and at the end of the section is a link called **Implementing the Scenario** which gives you instructions on how to create a VPC for that scenario. Follow these instructions to create a VPC.

> **Note**
> If you want your DB instance in the VPC to be publicly accessible, you must update the DNS information for the VPC by enabling the VPC attributes *DNS hostnames* and *DNS resolution*. For information about updating the DNS information for a VPC instance, see Updating DNS Support for Your VPC.

Once you have a VPC, there are five steps to create a DB instance in a VPC.

- Step 1: Creating a VPC (p. 567)
- Step 2: Adding Subnets to a VPC (p. 567)
- Step 3: Creating a DB Subnet Group (p. 567)
- Step 4: Creating a VPC Security Group (p. 568)
- Step 5: Creating a DB Instance in a VPC (p. 568)

Step 1: Creating a VPC

If your AWS account does not have a default VPC or if you want to create an additional VPC, follow the instructions for creating a VPC in the topic Step 1: Set Up the VPC and Internet Gateway in the Amazon VPC documentation.

Step 2: Adding Subnets to a VPC

Once you have created a VPC, you need to create a subnet in the VPC in at least two of the Availability Zones of the region. You will use these subnets when you create a DB subnet group. Note that if you have a default VPC, a subnet is automatically created for you in each Availability Zone in the region.

For instructions on how to create subnets in a VPC, see the topic Subnets in Your VPC in the Amazon VPC documentation.

Step 3: Creating a DB Subnet Group

A DB subnet group is a collection of subnets (typically private) that you create for a VPC and that you then designate for your DB instances. A DB subnet group allows you to specify a particular VPC when you create DB instances using the CLI or API; if you use the Amazon RDS console, you can just select the VPC and subnets you want to use. Each DB subnet group must have at least one subnet in at least two Availability Zones in the region.

> **Note**
> For a DB instance to be publicly accessible, the subnets in the DB subnet group must have an internet gateway. For more information about internet gateways for subnets, see Internet Gateways in the Amazon VPC documentation.

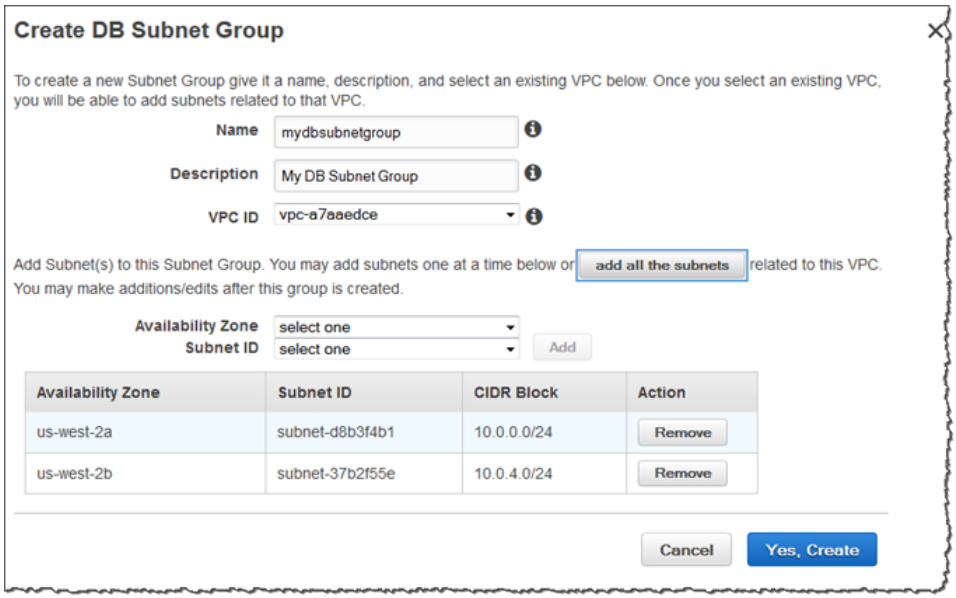
When you create a DB instance in a VPC, you must select a DB subnet group. Amazon RDS then uses that DB subnet group and your preferred Availability Zone to select a subnet and an IP address within that subnet. Amazon RDS creates and associates an Elastic Network Interface to your DB instance with that IP address. For Multi-AZ deployments, defining a subnet for two or more Availability Zones in a region allows Amazon RDS to create a new standby in another Availability Zone should the need arise. You need to do this even for Single-AZ deployments, just in case you want to convert them to Multi-AZ deployments at some point.

In this example, you create a DB subnet group and add the subnets you created for your VPC.

AWS Management Console

Create a DB subnet group

1. Open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Click **Subnet Groups** in the navigation pane on the left side of the console window.
3. Click the **Create DB Subnet Group** button.
4. Type the name of your DB subnet group in the **Name** text box.
5. Type a description for your DB subnet group in the **Description** text box.
6. Select the VPC that you created from the **VPC ID** drop-down list box.
7. In the **Add Subnet(s) to this Subnet Group** section, click the **add all the subnets** link.



8. Click **Yes, Create**.
9. Click **Close**.

   Your new DB subnet group appears in the DB subnet groups list on the RDS console. You can click the DB subnet group to see details, including all of the subnets associated with the group, in the details pane at the bottom of the window.

Step 4: Creating a VPC Security Group

Before you create your DB instance, you must create a VPC security group to associate with your DB instance. You can find out how to create a VPC security group in the Amazon VPC documentation.

Step 5: Creating a DB Instance in a VPC

In this step, you create a DB instance and use the VPC name, the DB subnet group, and the VPC security group you created in the previous steps.

> **Note**
> If you want your DB instance in the VPC to be publicly accessible, you must enable the VPC attributes *DNS hostnames* and *DNS resolution*. For information on updating the DNS information for a VPC instance, see Updating DNS Support for Your VPC.

Select the link below for your DB engine. In the **Launch DB Instance Wizard**, enter the VPC name, the DB subnet group, and the VPC security group you created in the previous steps when prompted.

- Creating a DB Instance Running the MySQL Database Engine (p. 129)
- Creating a DB Instance Running the Oracle Database Engine (p. 205)
- Creating a DB Instance Running the PostgreSQL Database Engine (p. 363)
- Creating a DB Instance Running the SQL Server Database Engine (p. 313)

Moving a DB Instance not in a VPC into a VPC

If you are moving a DB instance that is not in a VPC into a VPC, there are several steps you need to take. For example, if you want to move your DB instance that is not in a VPC from a db.t1 DB instance

class to a db.t2 DB instance class (which requires a VPC), you need to follow these steps. You must first create a VPC, create subnets in the VPC, create a DB subnet group that contains the subnets you created, and then create a VPC security group so you can access the DB instance in the VPC. Finally, you must create a DB snapshot of your DB instance, and then restore that DB snapshot using the VPC, DB subnet group, and VPC security group you created.

The following steps can be used to move a DB instance not in a VPC into a VPC:

- Step 1: Creating a VPC (p. 567)
- Step 2: Adding Subnets to a VPC (p. 567)
- Step 3: Creating a DB Subnet Group (p. 567)
- Step 4: Creating a VPC Security Group (p. 568)
- Step 5: Create a DB snapshot of the current DB instance that you want to move into a VPC

  Note that this operation requires that your database be offline since you are restoring a snapshot of your database. You will have to either stop writes to the database or apply the transaction logs after restoring the DB instance.

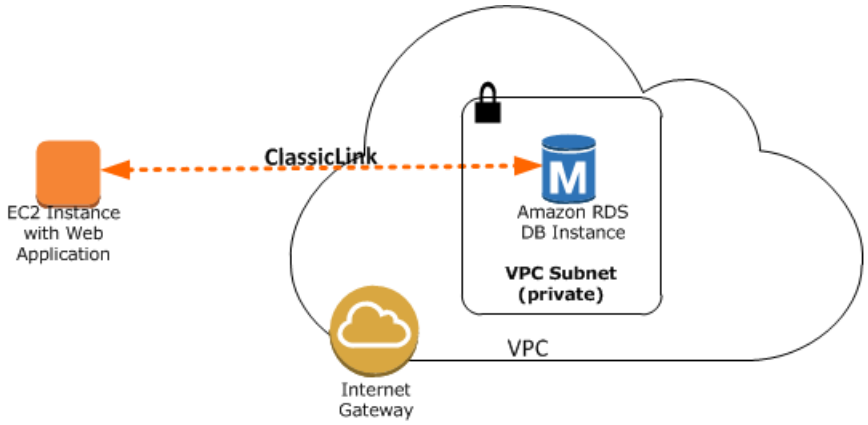
  To create a DB snapshot, follow the instructions in the section Creating a DB Snapshot (p. 499).
- Step 6: Restore the DB snapshot and specify the VPC, DB subnet group, and VPC security group you want to use.

  For more information about restoring from a DB snapshot, see Restoring From a DB Snapshot (p. 501)

  Note
  You cannot use a custom option group that is assigned to a DB instance not in a VPC when you move the DB instance into a VPC. Option groups are platform specific, so moving from a non-VPC to a VPC is a change in platform. Assign either the default option group to the DB instance, assign an option group that is used by other DB instances in the VPC you are moving to, or create a new option group and assign it to the DB instance.

Using ClassicLink to Connect an EC2-Classic Instance to a DB Instance in an Amazon VPC

You can communicate between an EC2 instance that is not in an Amazon Virtual Private Cloud (VPC) and an Amazon RDS DB instance that is in a VPC by using *ClassicLink*. Using this AWS feature, you can connect an EC2 instance to a logically isolated database where you define the IP address range and control the access control lists (ACLs) to manage network traffic. You don't have to use public IP addresses or tunneling to communicate with the DB instance in the VPC. This arrangement provides you with higher throughput and lower latency connectivity for inter-instance communications.



Note
The DB instance must be in a private subnet that is not open to the public (i.e. it cannot be set to publicly accessible).

To enable ClassicLink between a DB instance in a VPC and an EC2 instance not in a VPC

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.amazonaws.cn//vpc/.
2. In the left navigation pane, click **Your VPCs**.
3. In the VPC list, click the VPC used by the DB instance.
4. On the **Actions** menu, click **Enable ClassicLink**. In the confirmation dialog box, click **Yes, Enable**.
5. On the EC2 console, select the EC2 instance you want to connect to the DB instance in the VPC.
6. On the **Actions** menu, click **ClassicLink**, and then click **Link to VPC**.
7. On the **Link to VPC** page, click the security group you want to use, and then click **Link to VPC**.

An application on the EC2 instance can connect to the DB instance by using the RDS endpoint for the DB instance.

ClassicLink is available at no charge.

Monitoring Amazon RDS

There are several ways you can track the performance and health of a database or a DB instance. You can:

- Use the free Amazon CloudWatch service to monitor the performance and health of a DB instance.
- Subscribe to Amazon RDS events to be notified when changes occur with a DB instance, DB snapshot, DB parameter group, or DB security group.
- View, download, or watch database log files using the Amazon RDS console or Amazon RDS APIs. You can also query some database log files that are loaded into database tables.
- Use the AWS CloudTrail service to record AWS calls made by your AWS account. The calls are recorded in log files and stored in an Amazon S3 bucket.

Topics

- Viewing DB Instance Metrics (p. 572)
- Using Amazon RDS Event Notification (p. 575)
- Viewing Amazon RDS Events (p. 592)
- Amazon RDS Database Log Files (p. 594)
- Logging Amazon RDS API Calls Using AWS CloudTrail (p. 615)

Viewing DB Instance Metrics

Amazon RDS and CloudWatch are integrated so you can gather and monitor a variety of metrics. You can view CloudWatch metrics using the RDS console, CLI, or API.

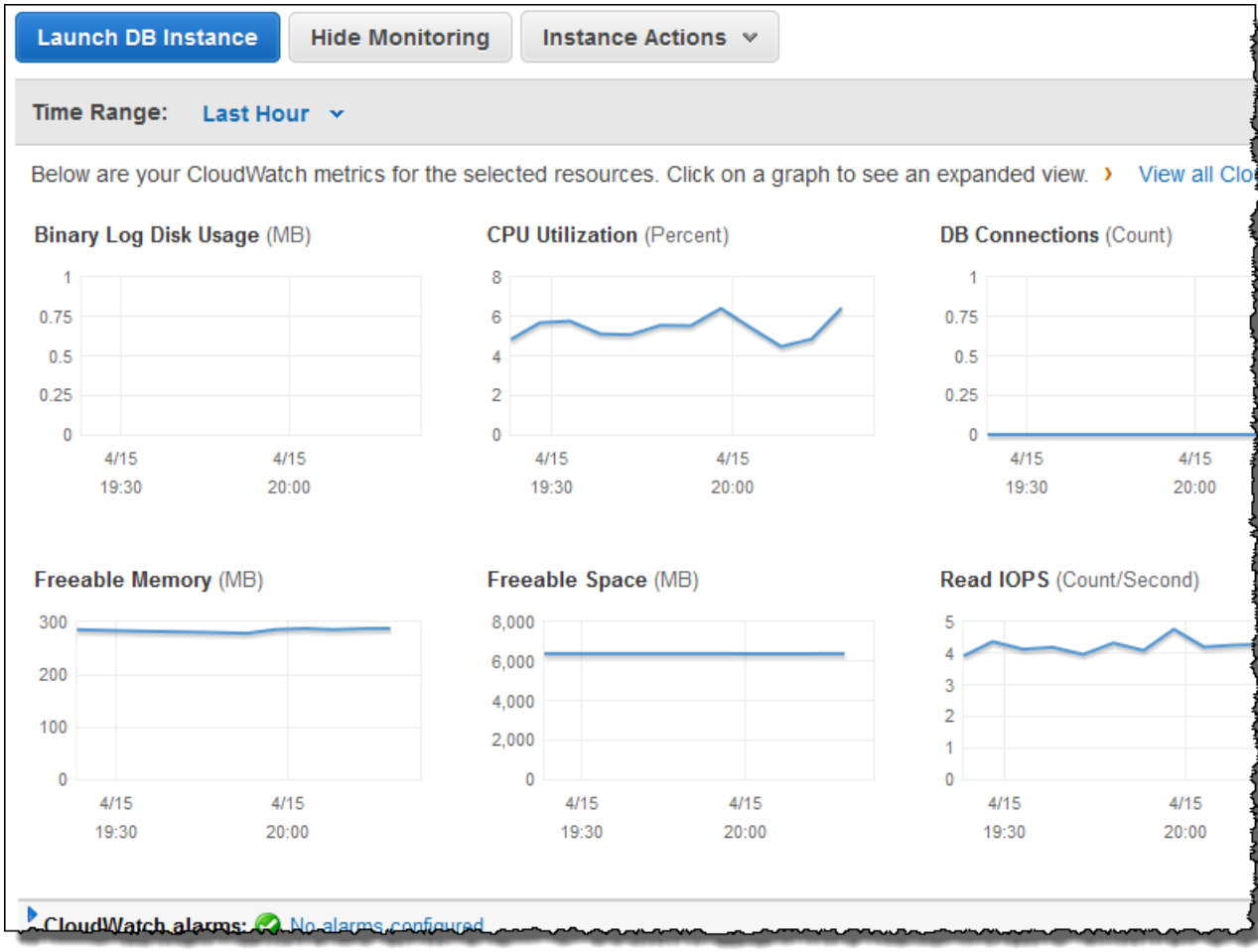
In the following example, you use CloudWatch to gather storage space statistics for an Amazon RDS DB instance for the past hour.

AWS Management Console

To view usage and performance statistics for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **DB Instances**.
3. Select the check box for the DB instance you want to monitor.
4. Click **Show Monitoring** at the top of the window.

   Graphs showing the metrics for the selected DB instance display in this tab.



Tip
You can use the **Time Range** drop-down list box to select the time range of the metrics represented by the graphs.
You can click on any of the graphs to bring up a more detailed view of the graph that allows you to apply additional metric-specific filters to the metric data.

CLI

Note
The following CLI example requires the CloudWatch command line tools. For more information on CloudWatch and to download the developer tools, go to the Amazon CloudWatch product page. Note that the `StartTime` and `EndTime` values supplied in this example are for illustrative purposes. You must substitute appropriate start and end time values for your DB instance.
For a complete list of Amazon RDS metrics, go to Amazon RDS Dimensions and Metrics in the Amazon CloudWatch Developer Guide.

To view usage and performance statistics for a DB instance

- Use the CloudWatch command **mon-get-stats** with the following parameters:

```
     PROMPT>mon-get-stats FreeStorageSpace --dimensions="DBInstanceIdenti
fier=mydbinstance" --statistics= Average
      --namespace="AWS/RDS" --start-time 2009-10-16T00:00:00 --end-time
2009-10-16T00:02:00
```

API

Note that the `StartTime` and `EndTime` values supplied in this example are for illustrative purposes. You must substitute appropriate start and end time values for your DB instance.

To view usage and performance statistics for a DB instance

- Call the CloudWatch API `GetMetricStatistics` with the following parameters:
  - *`Statistics.member.1`* = `Average`
  - *`Namespace`* = `AWS/RDS`
  - *`StartTime`* = `2009-10-16T00:00:00`
  - *`EndTime`* = `2009-10-16T00:02:00`
  - *`Period`* = `60`
  - *`MeasureName`* = `FreeStorageSpace`

Example

```
http://monitoring.amazonaws.com/
 ?SignatureVersion=2
 &Action=GetMetricStatistics
 &Version=2009-05-15
 &StartTime=2009-10-16T00:00:00
 &EndTime=2009-10-16T00:02:00
 &Period=60
 &Statistics.member.1=Average
 &Dimensions.member.1="DBInstanceIdentifier=mydbinstance"
 &Namespace=AWS/RDS
 &MeasureName=FreeStorageSpace
 &Timestamp=2009-10-15T17%3A48%3A21.746Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Related Topics

- Using Amazon RDS Event Notification (p. 575)
- Amazon RDS Database Log Files (p. 594)

Using Amazon RDS Event Notification

Topics

- Amazon RDS Event Categories and Event Messages (p. 576)
- Subscribing to Amazon RDS Event Notification (p. 580)
- Listing Your Amazon RDS Event Notification Subscriptions (p. 584)
- Modifying an Amazon RDS Event Notification Subscription (p. 585)
- Adding a Source Identifier to an Amazon RDS Event Notification Subscription (p. 587)
- Removing a Source identifier from an Amazon RDS Event Notification Subscription (p. 588)
- Listing the Amazon RDS Event Notification Categories (p. 589)
- Deleting an Amazon RDS Event Notification Subscription (p. 591)

Amazon RDS uses the Amazon Simple Notification Service (Amazon SNS) to provide notification when an Amazon RDS event occurs. These notifications can be in any notification form supported by Amazon SNS for an AWS region, such as an email, a text message, or a call to an HTTP endpoint.

Amazon RDS groups these events into categories that you can subscribe to so that you can be notified when an event in that category occurs. You can subscribe to an event category for a DB instance, DB snapshot, DB security group, or for a DB parameter group. For example, if you subscribe to the Backup category for a given DB instance, you will be notified whenever a backup-related event occurs that affects the DB instance. If you subscribe to a Configuration Change category for a DB security group, you will be notified when the DB security group is changed. You will also receive notification when an event notification subscription changes.

Event notifications are sent to the addresses you provide when you create the subscription. You may want to create several different subscriptions, such as one subscription receiving all event notifications and another subscription that includes only critical events for your production DB instances. You can easily turn off notification without deleting a subscription by setting the **Enabled** radio button to *No* in the Amazon RDS console or by setting the *Enabled* parameter to `false` using the CLI or Amazon RDS API.

> **Note**
> Amazon RDS event notifications using SMS text messages are currently available for topic ARNs and Amazon RDS resources in the US-East (Northern Virginia) Region. For more information on using text messages with SNS, see Sending and Receiving SMS Notifications Using Amazon SNS.

Amazon RDS uses the Amazon Resource Name (ARN) of an Amazon SNS topic to identify each subscription. The Amazon RDS console will create the ARN for you when you create the subscription. If you use the CLI or API, you have to create the ARN by using the Amazon SNS console or the Amazon SNS API when you create a subscription.

Billing for Amazon RDS event notification is through the Amazon Simple Notification Service (Amazon SNS). Amazon SNS fees apply when using event notification; for more information on Amazon SNS billing, see Amazon Simple Notification Service Pricing.

The process for subscribing to Amazon RDS event notification is as follows:

1. Create an Amazon RDS event notification subscription by using the Amazon RDS console, CLI, or API.
2. Amazon RDS sends an approval email or SMS message to the addresses you submitted with your subscription. To confirm your subscription, click the link in the notification you were sent.
3. When you have confirmed the subscription, the status of your subscription is updated in the Amazon RDS console's **My Event Subscriptions** section.
4. You will begin to receive event notifications.

The following section lists all categories and events that you can be notified of. It also provides information about subscribing to and working with Amazon RDS event subscriptions.

Amazon RDS Event Categories and Event Messages

Amazon RDS generates a significant number of events in categories that you can subscribe to using the Amazon RDS Console, CLI, or the API. Each category applies to a source type, which can be a DB instance, DB snapshot, DB security group, or DB parameter group.

Event categories for a DB instance source type include: Availability, Backup, Configuration Change, Creation, Deletion, Failover, Failure, Low Storage, Maintenance, Notification, Read Replica, Recovery, and Restoration.

Event categories for a DB snapshot source type include: Creation, Deletion, and Restoration.

Event categories for a DB security group source type include: Configuration Change and Failure.

The event category for a DB parameter group source type is Configuration Change.

The following table shows the event category and a list of events when a DB instance is the source type.

Categories and Events for the DB Instance Source Type

Category	Amazon RDS Event ID	Description
Availability	RDS-EVENT-0006	The DB Instance is restarting and will be unavailable until the restart is complete.
Availability	RDS-EVENT-0004	The DB Instance has shut down.
Availability	RDS-EVENT-0022	An error has occurred while restarting MySQL.
Backup	RDS-EVENT-0001	A backup of the DB instance has started.
Backup	RDS-EVENT-0002	A backup of the DB instance is complete.
Configuration Change	RDS-EVENT-0009	The DB instance has been added to a security group.
Configuration Change	RDS-EVENT-0024	The DB instance is being converted to a Multi-AZ DB instance.
Configuration Change	RDS-EVENT-0030	The DB instance is being converted to a Single-AZ DB instance.
Configuration Change	RDS-EVENT-0012	The DB instance class for this DB instance is being changed.
Configuration Change	RDS-EVENT-0018	The current storage settings for this DB instance is being changed.
Configuration Change	RDS-EVENT-0011	A parameter group for this DB instance has changed.
Configuration Change	RDS-EVENT-0028	Automatic backups for this DB instance have been disabled.
Configuration Change	RDS-EVENT-0032	Automatic backups for this DB instance have been enabled.

Category	Amazon RDS Event ID	Description
Configuration Change	RDS-EVENT-0033	There are [count] users that match the master user name. Users not tied to a specific host have been reset.
Configuration Change	RDS-EVENT-0025	The DB instance has been converted to a Multi-AZ DB instance.
Configuration Change	RDS-EVENT-0029	The DB instance has been converted to a Single-AZ DB instance.
Configuration Change	RDS-EVENT-0014	The DB instance class for this DB instance has changed.
Configuration Change	RDS-EVENT-0017	The storage settings for this DB instance has changed.
Configuration Change	RDS-EVENT-0010	The DB instance has been removed from a security group.
Configuration Change	RDS-EVENT-0016	The master password for the DB instance has been reset.
Configuration Change	RDS-EVENT-0067	An attempt to reset the master password for the DB instance has failed.
Creation	RDS-EVENT-0005	A DB instance is being created.
Deletion	RDS-EVENT-0003	The DB instance is being deleted.
Failover	RDS-EVENT-0034	Amazon RDS is not attempting a requested failover because a failover recently occurred on the DB instance.
Failover	RDS-EVENT-0013	A Multi-AZ failover that resulted in the promotion of a standby instance has started.
Failover	RDS-EVENT-0015	A Multi-AZ failover that resulted in the promotion of a standby instance is complete. It may take several minutes for the DNS to transfer to the new primary DB instance.
Failover	RDS-EVENT-0065	The instance has recovered from a partial failover.
Failover	RDS-EVENT-0050	A Multi-AZ activation has started after a successful instance recovery.
Failover	RDS-EVENT-0051	A Multi-AZ activation is complete. Your database should be accessible now.
Failure	RDS-EVENT-0031	The DB instance has failed. We recommend that you begin a point-in-time-restore for the DB instance.
Failure	RDS-EVENT-0036	The DB instance is in an incompatible network. Some of the specified subnet IDs are invalid or do not exist.
Failure	RDS-EVENT-0035	The DB instance has invalid parameters. For example, MySQL could not start because a memory-related parameter is set too high for this instance class, so the customer action would be to modify the memory parameter and reboot the DB instance.

Category	Amazon RDS Event ID	Description
Failure	RDS-EVENT-0058	Error while creating Statspack user account PERFSTAT. Please drop the account before adding the Statspack option.
Low Storage	RDS-EVENT-0007	The allocated storage for the DB instance has been exhausted. To resolve this issue, you should allocate additional storage for the DB instance. For more information, see the RDS FAQ.
Maintenance	RDS-EVENT-0026	Offline maintenance of the DB instance is taking place. The DB instance is currently unavailable.
Maintenance	RDS-EVENT-0027	Offline maintenance of the DB instance is complete. The DB instance is now available.
Maintenance	RDS-EVENT-0047	The read replication process has been stopped because replication errors have occurred for more than 30 days. The Read Replica is still accessible for read operations but cannot synchronize with the master instance. We recommend that you delete the Read Replica and create a new one. For information about troubleshooting a broken Read Replica, see Troubleshooting a MySQL Read Replica Problem (p. 483).
Notification	RDS-EVENT-0044	Operator-issued notification. For more information, see the event message.
Notification	RDS-EVENT-0048	Patching of the DB instance has been delayed.
Notification	RDS-EVENT-0049	A Multi-AZ failover has completed.
Notification	RDS-EVENT-0054	The MySQL storage engine you are using is not InnoDB, which is the recommended MySQL storage engine for Amazon RDS. For information about MySQL storage engines, see Amazon RDS Supported Storage Engines (p. 121).
Notification	RDS-EVENT-0055	The number of tables you have for your DB instance exceeds the recommended best practices for Amazon RDS. Please reduce the number of tables on your DB instance. For information about recommended best practices, see Amazon RDS Basic Operational Guidelines (p. 54).
Notification	RDS-EVENT-0056	The number of databases you have for your DB instance exceeds the recommended best practices for Amazon RDS. Please reduce the number of databases on your DB instance. For information about recommended best practices, see Amazon RDS Basic Operational Guidelines (p. 54).
Notification	RDS-EVENT-0064	The TDE key has been rotated. For more information about Oracle TDE, see Oracle Transparent Data Encryption (TDE) (p. 235). For more information about SQL Server TDE, see SQL Server Transparent Data Encryption (p. 350).

Category	Amazon RDS Event ID	Description
Read Replica	RDS-EVENT-0045	An error has occurred in the read replication process. For more information, see the event message. For information on troubleshooting Read Replica errors, see Troubleshooting a MySQL Read Replica Problem (p. 483).
Read Replica	RDS-EVENT-0046	The Read Replica has resumed replication. This message appears when you first create a Read Replica, or as a monitoring message confirming that replication is functioning properly. If this message follows an RDS-EVENT-0045 notification, then replication has resumed following an error or after replication was stopped.
Read Replica	RDS-EVENT-0057	Replication on the Read Replica was terminated.
Read Replica	RDS-EVENT-0062	Replication on the Read Replica was manually stopped.
Read Replica	RDS-EVENT-0063	Replication on the Read Replica was reset.
Recovery	RDS-EVENT-0020	Recovery of the DB instance has started. Recovery time will vary with the amount of data to be recovered.
Recovery	RDS-EVENT-0021	Recovery of the DB instance is complete.
Recovery	RDS-EVENT-0023	A manual backup has been requested but Amazon RDS is currently in the process of creating a DB snapshot. Submit the request again after Amazon RDS has completed the DB snapshot.
Recovery	RDS-EVENT-0052	Recovery of the Multi-AZ instance has started. Recovery time will vary with the amount of data to be recovered.
Recovery	RDS-EVENT-0053	Recovery of the Multi-AZ instance is complete.
Recovery	RDS-EVENT-0066	The SQL Server DB instance is re-establishing its mirror. Performance will be degraded until the mirror is reestablished. A database was found with non-FULL recovery model. The recovery model was changed back to FULL and mirroring recovery was started. (<db-name>: <recovery model found>[,…])”
Restoration	RDS-EVENT-0008	The DB instance has been restored from a DB snapshot.
Restoration	RDS-EVENT-0019	The DB instance has been restored from a point-in-time backup.
Security	RDS-EVENT-0068	The CloudHSM partition password was decrypted by the system.

The following table shows the event category and a list of events when a DB parameter group is the source type.

Categories and Events for the DB Parameter Group Source Type

Category	RDS Event ID	Description
Configuration Change	RDS-EVENT-0037	The parameter group was modified.

The following tables shows the event category and a list of events when a DB security group is the source type.

Categories and Events for the DB Security Group Source Type

Category	RDS Event ID	Description
Configuration Change	RDS-EVENT-0038	The security group has been modified.
Failure	RDS-EVENT-0039	The Amazon EC2 security group owned by [user] does not exist; authorization for the security group has been revoked.

The following tables shows the event category and a list of events when a DB snapshot is the source type.

Categories and Events for the DB Snapshot Source Type

Category	RDS Event ID	Description
Creation	RDS-EVENT-0040	A DB snapshot is being created.
Creation	RDS-EVENT-0042	A DB snapshot has been created.
Deletion	RDS-EVENT-0041	A DB snapshot has been deleted.
Notification	RDS-EVENT-0059	Started the copy of DB snapshot [DB snapshot name] from source region [region name].
Failure	RDS-EVENT-0061	The copy of a DB snapshot failed.
Notification	RDS-EVENT-0060	Finished the copy of DB snapshot [DB snapshot name] from source region [region name] in [time] minutes.
Restoration	RDS-EVENT-0043	A DB instance is being restored from a DB snapshot.

Subscribing to Amazon RDS Event Notification

You can create an Amazon RDS event notification subscription so you can be notified when an event occurs for a given DB instance, DB snapshot, DB security group, or DB parameter group. The simplest way to create a subscription is with the RDS console. If you choose to create event notification subscriptions using the CLI or API, you must create an Amazon Simple Notification Service topic and subscribe to that topic with the Amazon SNS console or Amazon SNS API. You will also need to retain the Amazon Resource Name (ARN) of the topic because it is used when submitting CLI commands or API actions. For information on creating an SNS topic and subscribing to it, see Getting Started with Amazon SNS.

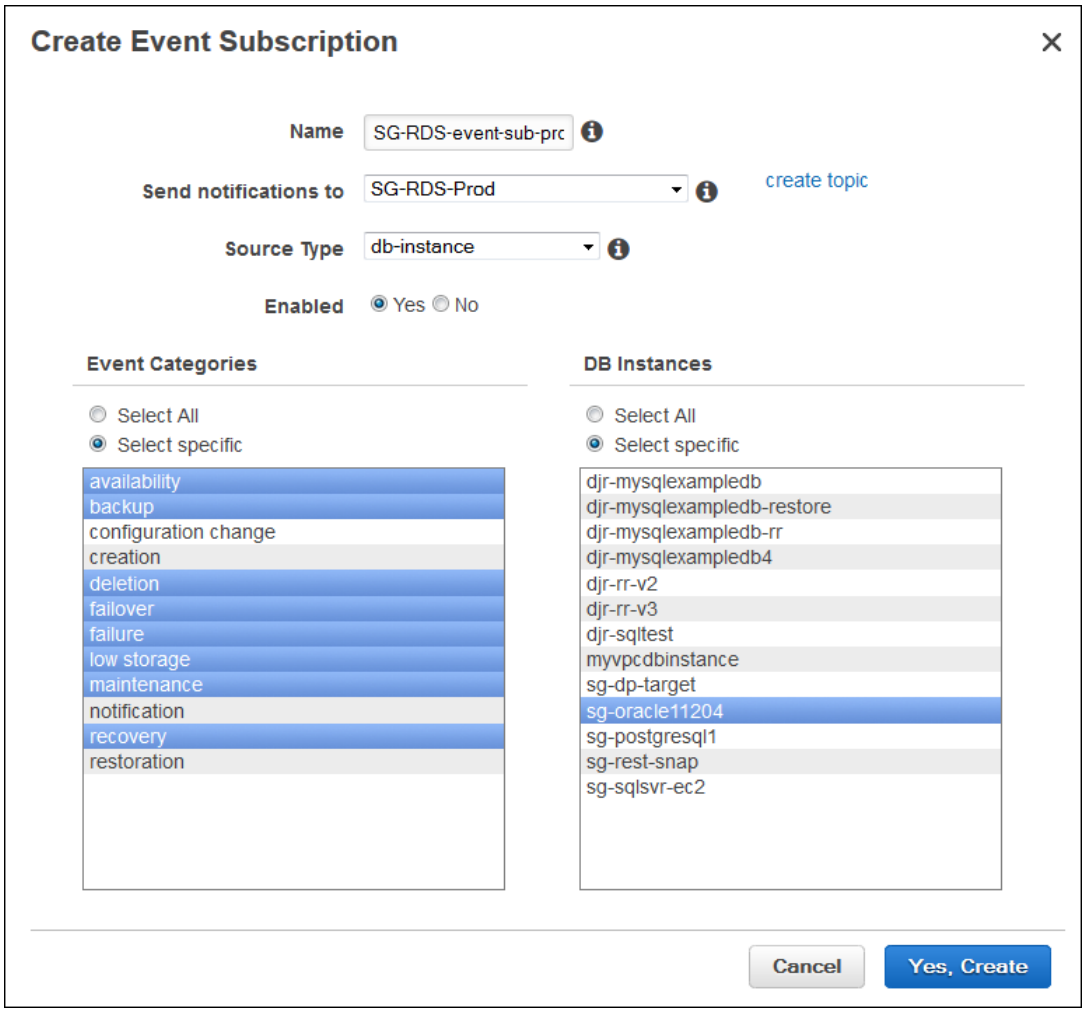
You can specify the type of source you want to be notified of and the Amazon RDS source that triggers the event. These are defined by the **SourceType** (type of source) and the **SourceIdentifier** (the Amazon RDS source generating the event). If you specify both the **SourceType** and **SourceIdentifier**, such as `SourceType = db-instance` and `SourceIdentifier = myDBInstance1`, you will receive all the

DB_Instance events for the specified source. If you specify a **SourceType** but do not specify a **SourceIdentifier**, you will receive notice of the events for that source type for all your Amazon RDS sources. If you do not specify either the **SourceType** nor the **SourceIdentifier**, you will be notified of events generated from all Amazon RDS sources belonging to your customer account.

AWS Management Console

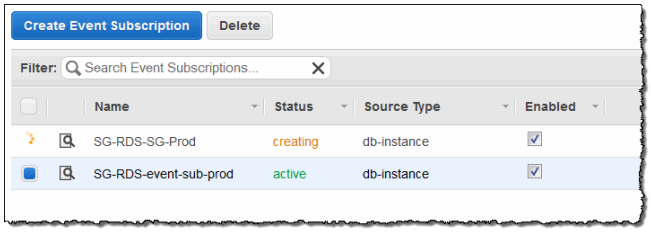
To subscribe to RDS event notification

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the Amazon RDS Console navigation pane, click **Event Subscriptions**.
3. In the **Event Subscriptions** pane, click **Create Event Subscription**.
4. In the **Create Event Subscription** dialog box, do the following:
    a. Type a name for the event notification subscription in the **Name** text box.
    b. Select an existing Amazon SNS Amazon Resource Name (ARN) for an Amazon SNS topic in the **Send notifications to** dropdown menu or click **create topic** to enter the name of a topic and a list of recipients.
    c. Select a source type from the **Source Type** dropdown menu.
    d. Select **Yes** to enable the subscription. If you want to create the subscription but to not have notifications sent yet, select **No**.
    e. Depending on the source type you selected, select the event categories and sources you want to receive event notifications for.



f. Click **Yes, Create**.

5. The Amazon RDS console indicates that the subscription is being created.



CLI

To subscribe to RDS Event Notification

- Use the `rds-create-event-subscription` command.

API

To subscribe to Amazon RDS Event Notification

- Call `CreateEventSubscription`.

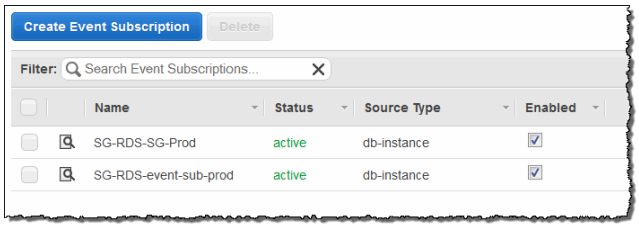
Listing Your Amazon RDS Event Notification Subscriptions

You can list your current Amazon RDS event notification subscriptions.

AWS Management Console

To list your current Amazon RDS event notification subscriptions

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the Amazon RDS Console navigation pane, click **Event Subscriptions**. The Event Subscriptions pane shows all your event notification subscriptions.



CLI

To list your current Amazon RDS event notification subscriptions

- Use the `rds-describe-event-subscriptions` command.

API

To list your current Amazon RDS event notification subscriptions

- Call `DescribeEventSubscriptions`.

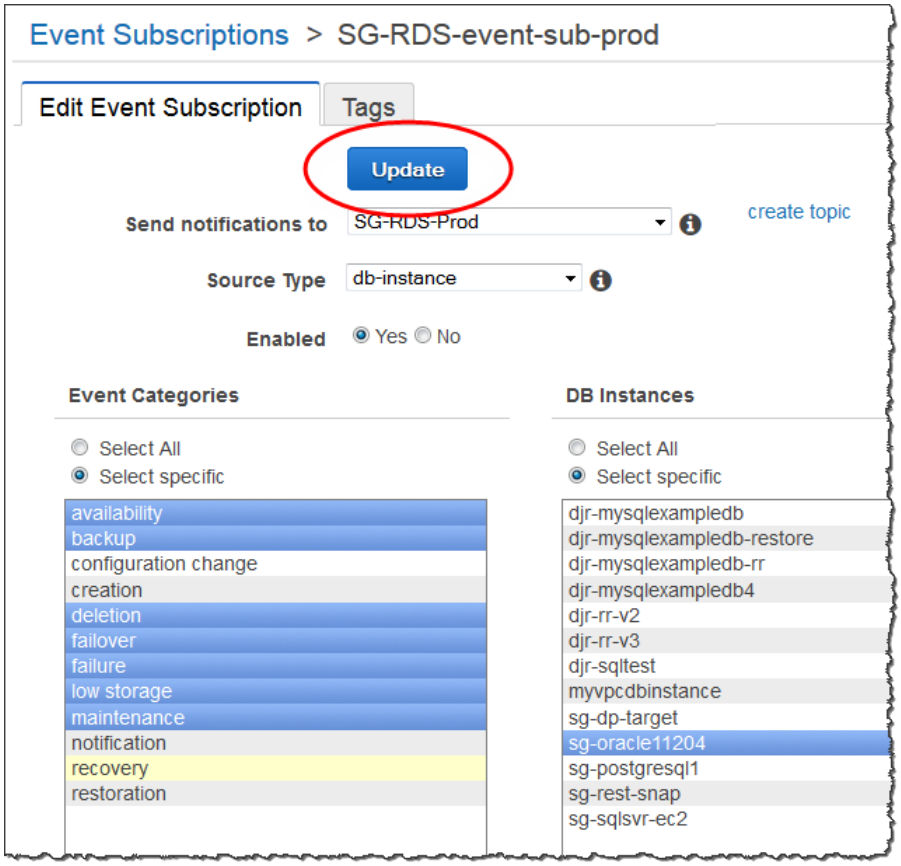
Modifying an Amazon RDS Event Notification Subscription

After you have created a subscription, you can change the subscription name, source identifier, categories, or topic ARN.

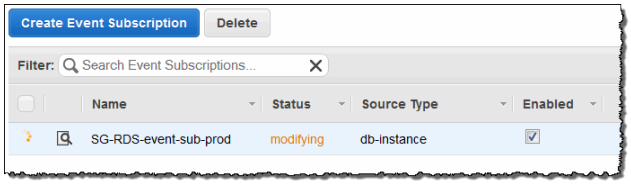
AWS Management Console

To modify an Amazon RDS event notification subscription

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the Amazon RDS Console navigation pane, click **Event Notification**.
3. In the **DB Event Notifications** pane, select the subscription that you want to modify.
4. Make your changes to the subscription in the lower pane.



5. Click **Update**. The Amazon RDS console indicates that the subscription is being modified.



CLI

To modify an Amazon RDS event notification subscription

- Use the `rds-modify-event-subscription` command.

API

To modify an Amazon RDS Event

- Call `ModifyEventSubscription`.

Adding a Source Identifier to an Amazon RDS Event Notification Subscription

You can add a source identifier (the Amazon RDS source generating the event) to an existing subscription.

AWS Management Console

To add a source identifier to an Amazon RDS event notification subscription

- You can easily add or remove source identifiers using the Amazon RDS console by selecting or deselecting them when modifying a subscription. See the topic Modifying an Amazon RDS Event Notification Subscription (p. 585) for more information.

CLI

To add a source identifier to an Amazon RDS event notification subscription

- Use the `rds-add-source-identifier-to-subscription` command.

API

To add a source identifier to an Amazon RDS event notification subscription

- Call `AddSourceIdentifierToSubscription`.

Removing a Source identifier from an Amazon RDS Event Notification Subscription

You can remove a source identifier (the Amazon RDS source generating the event) from a subscription if you no longer want to be notified of events for that source.

AWS Management Console

To remove a source identifier from an Amazon RDS event notification subscription

- You can easily add or remove source identifiers using the Amazon RDS console by selecting or deselecting them when modifying a subscription. See the topic Modifying an Amazon RDS Event Notification Subscription (p. 585) for more information.

CLI

To remove a source identifier from an Amazon RDS event notification subscription

- Use the `rds-remove-source-identifier-from-subscription` command.

API

To remove a source identifier from an Amazon RDS event notification subscription

- Call `RemoveSourceIdentifierFromSubscription`.

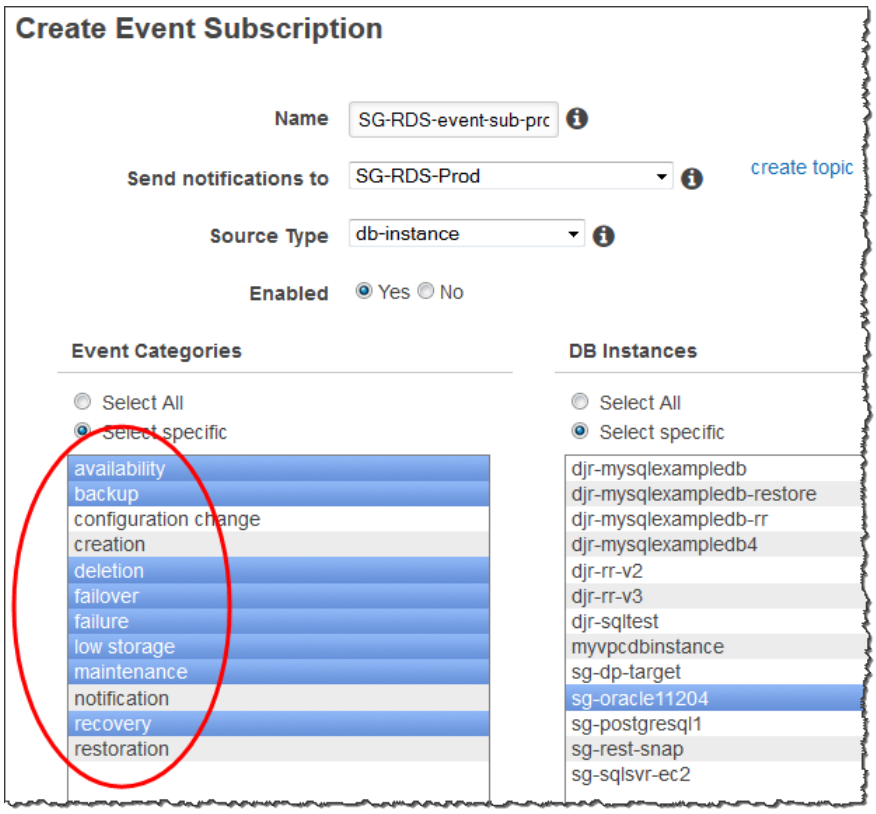
Listing the Amazon RDS Event Notification Categories

All events for a resource type are grouped into categories. To view the list of categories available, use the following procedures.

AWS Management Console

To list the Amazon RDS event notification categories

- When you create or modify an event notification subscription, the event categories are displayed in the Amazon RDS console. See the topic Modifying an Amazon RDS Event Notification Subscription (p. 585) for more information.



CLI

To list the Amazon RDS event notification categories

- Use the `rds-describe-event-categories` command.

API

To list the Amazon RDS event notification categories

- Call `DescribeEventCategories`.

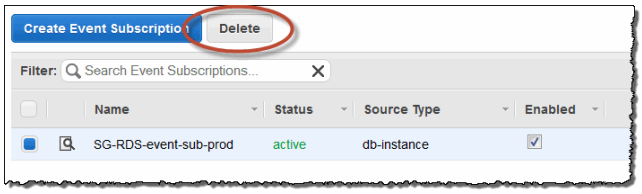
Deleting an Amazon RDS Event Notification Subscription

You can delete a subscription when you no longer need it. All subscribers to the topic will no longer receive event notifications specified by the subscription.

AWS Management Console

To delete an Amazon RDS event notification subscription

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the Amazon RDS Console navigation pane, click **DB Event Subscriptions**.
3. In the **My DB Event Subscriptions** pane, click the subscription that you want to delete.
4. Click **Delete**.
5. The Amazon RDS console indicates that the subscription is being deleted.



CLI

To delete an Amazon RDS event notification subscription

- Use the `rds-delete-event-subscription` command.

API

To delete an Amazon RDS event notification subscription

- Call `DeleteEventSubscription`.

Viewing Amazon RDS Events

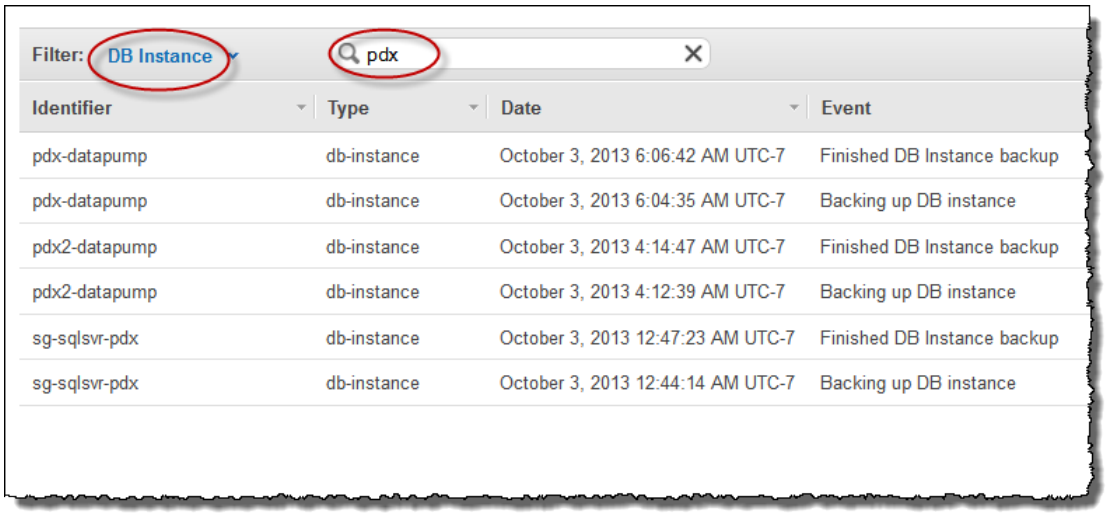
Amazon RDS keeps a record of events that relate to your DB Instances, DB Snapshots, DB Security Groups, and DB Parameter Groups. This information includes the date and time of the event, the source name and source type of the event, and a message associated with the event. You can easily retrieve events for your RDS resources through the AWS Management Console, the **rds-describe-events** CLI command, or the **DescribeEvents** API.

In this example, you view all Amazon RDS events for the past 24 hours (specified in seconds). Events are retained for 14 days.

AWS Management Console

To view all Amazon RDS instance events for the past 24 hours

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Click **Events** in the navigation list on the left side of the window. The available events appear in a list.
3. You can use the **Filter** drop-down list box to filter the events by type, and you can use the text box to the right of the **Filter** drop-down list box to further filter your results. For example, the following screen shot shows a list of events filtered by the DB Instance event type and containing the letters "pdx."



CLI

To view all Amazon RDS instance events for the past 24 hours

- Use the command `rds-describe-events` with the following parameters to view all Amazon RDS events for the past 24 hours.

```
PROMPT>rds-describe-events --duration 1440
```

API

To view all Amazon RDS instance events for the past 24 hours

- Call `DescribeEvents` with the following parameters:
  - *`Duration`* = `1440`

Example

```
https://rds.amazonaws.com/
 ?Action=DescribeEvents
 &Duration=1440
 &MaxRecords=100
 &Version=2012-01-15
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2012-01-22T20%3A00%3A44.420Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Related Topics

- Using Amazon RDS Event Notification (p. 575)

Amazon RDS Database Log Files

Topics

- MySQL Database Log Files (p. 594)
- Oracle Database Log Files (p. 598)
- SQL Server Database Log Files (p. 602)
- PostgreSQL Database Log Files (p. 603)
- Viewing and Listing Database Log Files (p. 605)
- Downloading a Database Log File (p. 609)
- Watching a Database Log File (p. 611)

You can view, download, and watch database logs using the Amazon RDS console, the Command Line Interface (CLI), or the Amazon RDS API. For example, you can view, download, or watch the error log, slow query log, and general logs for MySQL. You can also view MySQL logs by directing the logs to a database table in the main database and querying that table. Note that viewing, watching, or downloading transaction logs is not supported.

There are five ways to access database log files.

- View: You can view the contents of a log file by using the Amazon RDS console. You can also run the **rds-describe-db-log-file** command or call the **DescribeDBLogFiles** API action to list the log files that are available for a DB instance.
- Watch: You can view real-time updates to log files by using the Amazon RDS console. You can also run the **rds-watch-db-logfile** command or call or the **DownloadDBLogFilePortion** API action to monitor a database log file and poll to retrieve the most recent log file contents.
- Download: You can download the contents of a log file using the Amazon RDS console. You can also run the **rds-download-db-logfile** command to download the contents of a log file.
- Query: You can direct the MySQL slow query log and general log to a database table and run queries against the table to get the contents of the log file. Enabling the logs to be written to a database table can cause performance degradation.
- Binary: You can use the MySQL mysqlbinlog utility to download or stream a binary log to your local computer.

> **Note**
> If you cannot view the list of log files for an existing Oracle DB instance, reboot the instance to view the list.

MySQL Database Log Files

You can monitor the MySQL error log, slow query log, and the general log directly through the Amazon RDS console, Amazon RDS API, Amazon RDS CLI, or AWS SDKs. You can use the mysqlbinlog utility to download or stream a binary log. The MySQL error log is generated by default; you can generate the slow query and general logs by setting parameters in your DB parameter group. Amazon RDS rotates all of the MySQL log files, the intervals for each type are given below.

Binary Logging Format

MySQL on Amazon RDS supports both the *row-based* and *mixed* binary logging formats for MySQL version 5.6. The default binary logging format is mixed. For DB instances running MySQL versions 5.1 and 5.5, only mixed binary logging is supported. For details on the different MySQL binary log formats, see Binary Logging Formats in the *MySQL Reference Manual*.

Important
Setting the binary logging format to row-based can result in very large binary log files. Large binary log files reduce the amount of storage available for a DB instance and can increase the amount of time to perform a restore operation of a DB instance.

To set the MySQL binary logging format:

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. Click Parameter Groups in the left pane.
3. For the **default.mysql5.6** DB parameter group, click the **Go to Details Page** icon.
4. Click the Edit Parameters button to modify the parameters in the DB parameter group.
5. Set the `binlog_format` parameter to the binary logging format of your choice (**MIXED** or **ROW**).
6. Click the Save Changes button to save the updates to the DB parameter group.

Important
Changing the default.mysql5.6 DB parameter group affects all MySQL version 5.6 DB instances that use that parameter group. If you want to specify different binary logging formats for different MySQL 5.6 DB instances in a region, you will need to create your own DB parameter group that identifies the different logging format and assign that DB parameter group to the intended DB instances.

For more information on DB parameter groups, see Working with DB Parameter Groups (p. 523).

Accessing MySQL Error Logs

The MySQL error log is written to the `mysql-error.log` file. You can view `mysql-error.log` by using the Amazon RDS console or by retrieving the log using the Amazon RDS API, Amazon RDS CLI, or AWS SDKs. `mysql-error.log` is flushed every 5 minutes, and its contents are appended to `mysql-error-running.log`. The `mysql-error-running.log` file is then rotated every hour and the hourly files generated during the last 24 hours are retained. Each log file has the hour it was generated (in UTC) appended to its name. The log files also have a timestamp that helps you determine when the log entries were written.

MySQL writes to the error log only on startup, shutdown, and when it encounters errors. A DB instance can go hours or days without new entries being written to the error log. If you see no recent entries, it's because the server did not encounter an error that would result in a log entry.

Accessing the MySQL Slow Query and General Logs

The MySQL slow query log and the general log can be written to a file or a database table by setting parameters in your DB parameter group. For information about creating and modifying a DB parameter group, see Working with DB Parameter Groups (p. 523). You must set these parameters before you can view the slow query log or general log in the Amazon RDS console or by using the Amazon RDS API, Amazon RDS CLI, or AWS SDKs.

You can control MySQL logging by using the parameters in this list:

- *`slow_query_log`*: To create the slow query log, set to 1. The default is 0.
- *`general_log`*: To create the general log, set to 1. The default is 0.
- *`long_query_time`*: To prevent fast-running queries from being logged in the slow query log, specify a value for the shortest query execution time to be logged, in seconds. The default is 10 seconds, the minimum is 0. If log_output = FILE, you can specify a floating point value that goes to microsecond resolution. If log_output = TABLE, you must specify an integer value with second resolution. Only queries whose execution time exceeds the *`long_query_time`* value are logged. For example, setting

long_query_time to 0.1 prevents any query that runs for less than 100 milliseconds from being logged.

- *log_queries_not_using_indexes*: To log all queries that do not use an index to the slow query log, set to 1. The default is 0. Queries that do not use an index are logged even if their execution time is less than the value of the *long_query_time* parameter.
- *log_output option*: You can specify one of the following options for the `log_output` parameter.
  - **TABLE** (default)– Write general queries to the `mysql.general_log` table, and slow queries to the `mysql.slow_log` table.
  - **FILE**– Write both general and slow query logs to the file system. Log files are rotated hourly.
  - **NONE**– Disable logging.

When logging is enabled, Amazon RDS rotates table logs or deletes log files at regular intervals. This measure is a precaution to reduce the possibility of a large log file either blocking database use or affecting performance. `FILE` and `TABLE` logging approach rotation and deletion as follows:

- When `FILE` logging is enabled, log files are examined every hour and log files older than 24 hours are deleted. If the remaining combined log file size after the deletion exceeds a threshold of 2% of a DB instance's allocated space, then the largest log files are deleted until the log file size no longer exceeds the threshold.
- When `TABLE` logging is enabled, log tables are rotated every 24 hours if the space used by the table logs is more than 20 percent of the allocated storage space or the size of all logs combined is greater than 10 GB. If the amount of space used for a DB instance is greater than 90 percent of the DB instance's allocated storage space, then the thresholds for log rotation are reduced. Log tables are then rotated if the space used by the table logs is more than 10 percent of the allocated storage space or the size of all logs combined is greater than 5 GB.

  When log tables are rotated, the current log table is copied to a backup log table and the entries in the current log table are removed. If the backup log table already exists, then it is deleted before the current log table is copied to the backup. You can query the backup log table if needed. The backup log table for the `mysql.general_log` table is named `mysql.general_log_backup`. The backup log table for the `mysql.slow_log` table is named `mysql.slow_log_backup`.

  You can rotate the `mysql.general_log` table by calling the `mysql.rds_rotate_general_log` procedure. You can rotate the `mysql.slow_log` table by calling the `mysql.rds_rotate_slow_log` procedure.

  Table logs are rotated during a database version upgrade.

Amazon RDS records both `TABLE` and `FILE` log rotation in an Amazon RDS event and sends you a notification.

To work with the logs from the Amazon RDS console, Amazon RDS API, Amazon RDS CLI, or AWS SDKs, set the *log_output* parameter to FILE. Like the MySQL error log, these log files are rotated hourly. The log files that were generated during the previous 24 hours are retained.

For more information about the slow query and general logs, go to the following topics in the MySQL documentation:

- The Slow Query Log
- The General Query Log

Accessing File-Based MySQL Logs

You can access the file-based MySQL logs, such as the general or slow query logs, using the Amazon RDS features for working with logs:

- For information about listing log files and viewing the contents of a log, see Viewing and Listing Database Log Files (p. 605).
- For information about downloading a log file, see Downloading a Database Log File (p. 609).
- For information about watching real time updates to a log file, see Watching a Database Log File (p. 611).

Log File Size

The MySQL slow query log, error log, and the general log file sizes are constrained to no more than 2% of the allocated storage space for a DB instance. To maintain this threshold, logs are automatically rotated every hour and log files older than 24 hours are removed. If the combined log file size exceeds the threshold after removing old log files, then the largest log files are deleted until the log file size no longer exceeds the threshold.

For MySQL version 5.6.20 and later, there is a size limit on BLOBs written to the redo log. To account for this limit, ensure that the `innodb_log_file_size` parameter for your MySQL DB instance is 10 times larger than the largest BLOB data size found in your tables, plus the length of other variable length fields (`VARCHAR`, `VARBINARY`, `TEXT`) in the same tables. For information on how to set parameter values, see Working with DB Parameter Groups (p. 523). For information on the redo log BLOB size limit, go to Changes in MySQL 5.6.20.

Managing Table-Based MySQL Logs

You can direct the general and slow query logs to tables on the DB instance by creating a DB parameter group and setting the *`log_output`* server parameter to `TABLE`. General queries are then logged to the `mysql.general_log` table, and slow queries are logged to the `mysql.slow_log` table. You can query the tables to access the log information. Enabling this logging increases the amount of data written to the database, which can degrade performance.

Both the general log and the slow query logs are disabled by default. In order to enable logging to tables, you must also set the *`general_log`* and *`slow_query_log`* server parameters to `1`.

Log tables will keep growing until the respective logging activities are turned off by resetting the appropriate parameter to `0`. A large amount of data often accumulates over time, which can use up a considerable percentage of your allocated storage space. Amazon RDS does not allow you to truncate the log tables, but you can move their contents. Rotating a table saves its contents to a backup table and then creates a new empty log table. You can manually rotate the log tables with the following command line procedures, where the command prompt is indicated by `PROMPT>`:

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

To completely remove the old data and reclaim the disk space, call the appropriate procedure twice in succession.

Accessing MySQL 5.6 Binary Logs

You can use the mysqlbinlog utility to download or stream binary logs from Amazon RDS instances running MySQL 5.6. The binary log is downloaded to your local computer, where you can perform actions

such as replaying the log using the mysql utility. For more information about using the mysqlbinlog utility, go to Using mysqlbinlog to Back Up Binary Log Files.

To run the mysqlbinlog utility against an Amazon RDS instance, use the following options:

- Specify the `--read-from-remote-server` option.
- `--host`: Specify the DNS name from the endpoint of the instance.
- `--port`: Specify the port used by the instance.
- `--user`: Specify a MySQL user that has been granted the replication slave permission.
- `--password`: Specify the password for the user, or omit a password value so the utility will prompt you for a password.
- To have the file downloaded in binary format, specify the `--raw` option.
- `--result-file`: Specify the local file that will receive the raw output.
- Specify the names of one or more binary log files. To get a list of the available logs, use the SQL command SHOW BINARY LOGS.
- To stream the binary log files, specify the `--stop-never` option.

For more information about mysqlbinlog options, go to mysqlbinlog - Utility for Processing Binary Log Files.

For example:

```
mysqlbinlog --read-from-remote-server --host=MySQL56Instance1.cg034hpkmmjt.re
gion.rds.amazonaws.com --port=3306  --user ReplUser --password --raw --result-
file=/tmp/ binlog.00098
```

Amazon RDS normally purges a binary log as soon as possible, but the binary log must still be available on the instance to be accessed by mysqlbinlog. To specify the number of hours for RDS to retain binary logs, use the `mysql.rds_set_configuration` stored procedure and specify a period with enough time for you to download the logs. After you set the retention period, monitor storage usage for the DB instance to ensure that the retained binary logs do not take up too much storage.

This example sets the retention period to 1 day:

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

To display the current setting, use the `mysql.rds_show_configuration` stored procedure:

```
call mysql.rds_show_configuration;
```

Oracle Database Log Files

You can access Oracle alert logs, audit files, and trace files by using the Amazon RDS console or APIs. These files are retained for seven days by default. Note that the Oracle database engine may rotate these logs if they get very large. If you want to retain audit or trace files for a longer period, you should download them. Storing the files locally reduces your Amazon RDS storage costs and makes more space available for your data.

The Oracle audit files provided are the standard Oracle auditing files. While Fine Grained Auditing is a supported feature, log access does not provide access to Fine-Grained Auditing (FGA) events stored in the SYS.FGA_LOG$ table and that are accessible through the DBA_FGA_AUDIT_TRAIL view.

The **DescribeDBLogFiles** API action that lists the Oracle log files that are available for a DB instance ignores the `MaxRecords` parameter and returns up to 1000 records.

Switching Online Log files

In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges. You can use the following Amazon RDS-specific implementations to switch online log files.

Oracle Method	Amazon RDS Method
`alter system switch logfile;`	`exec rdsadmin.rdsadmin_util.switch_logfile;`

Working with Oracle Trace Files

This section describes Amazon RDS-specific procedures to create, refresh, access, and delete trace files.

Listing Files

Two procedures are available to allow access to any file within the `background_dump_dest`. The first method refreshes a view containing a listing of all files currently in the `background_dump_dest`:

```
exec rdsadmin.manage_tracefiles.refresh_tracefile_listing;
```

Once the view is refreshed, use the following view to access the results.

```
rdsadmin.tracefile_listing
```

An alternative to the previous process (available beginning with version 11.2.0.3.v1) is to use "from table" to stream non-table data in a table-like format to list DB directory contents:

```
SELECT * FROM table(rdsadmin.rds_file_util.listdir('BDUMP'));
```

The following query shows text of a log file:

```
SELECT text FROM table(rdsad
min.rds_file_util.read_text_file('BDUMP','alert_xxx.log'));
```

Generating Trace Files

Since there are no restrictions on `alter session`, many standard methods to generate trace files in Oracle remain available to an Amazon RDS DB instance. The following procedures are provided for trace files that require greater access.

Hanganalyze

Oracle Method	Amazon RDS Method
`oradebug hanganalyze 3`	`exec rdsadmin.manage_tracefiles.hanganalyze;`

System State Dump

Oracle Method	Amazon RDS Method
`oradebug dump systemstate 266`	`exec rdsadmin.manage_tracefiles.dump_system-state;`

Retrieving Trace Files

You can retrieve any trace file in `background_dump_dest` using a standard SQL query of an Amazon RDS managed external table. To use this method, you must execute the procedure to set the location for this table to the specific trace file.

For example, you can use the `rdsadmin.tracefile_listing` view mentioned above to list the all of the trace files on the system. You can then set the tracefile_table view to point to the intended trace file using the following procedure:

```
exec
rdsadmin.manage_tracefiles.set_tracefile_table_location('CUST01_ora_3260_SYSTEMSTATE.trc');
```

The following example creates an external table in the current schema with the location set to the file provided. The contents can be retrieved into a local file using a SQL query.

```
# eg: send the contents of the tracefile to a local file:
sql customer_dba/password@cust01 << EOF > /tmp/systemstatedump.txt
select * from tracefile_table;
EOF
```

Purging Trace Files

Trace files can accumulate and consume disk space. Amazon RDS purges trace files by default and log files that are older than seven days. You can view and set the trace file retention period using the show_configuration procedure. Note that you should run the command `SET SERVEROUTPUT ON` so that you can view the configuration results.

The following example shows the current trace file retention period, and then sets a new trace file retention period.

```
# Show the current tracefile retention
SQL> exec rdsadmin.rdsadmin_util.show_configuration;
NAME:tracefile retention
VALUE:10080
DESCRIPTION:tracefile expiration specifies the duration in minutes before
tracefiles in bdump are automatically deleted.

# Set the tracefile retention to 24 hours:
```

```
SQL> exec rdsadmin.rdsadmin_util.set_configuration('tracefile retention',1440);

#show the new tracefile retention
SQL> exec rdsadmin.rdsadmin_util.show_configuration;
NAME:tracefile retention
VALUE:1440
DESCRIPTION:tracefile expiration specifies the duration in minutes before
tracefiles in bdump are automatically deleted.
```

In addition to the periodic purge process, you can manually remove files from the `background_dump_dest`. The following example shows how to purge all files older than five minutes.

```
exec rdsadmin.manage_tracefiles.purge_tracefiles(5);
```

You can also purge all files that match a specific pattern (do not include the file extension such as .trc). The following example shows how to purge all files that start with "SCHPOC1_ora_5935".

```
exec rdsadmin.manage_tracefiles.purge_tracefiles('SCHPOC1_ora_5935');
```

Retrieving Archived Redo Logs

If you are using Oracle Database 11.2.0.2.v7 or later, you can retain archived redo logs and use log miner (DBMS_LOGMNR) to retrieve log information.

For example, the following command retains redo logs for 24 hours:

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
```

Because these logs are retained on your DB instance, you need to ensure that you have enough storage available on your instance to accommodate the log files. To see how much space you have used in the last "X" hours, use the following query, replacing "X" with the number of hours.

```
select sum(blocks * block_size) bytes from v$archived_log where
next_time>=sysdate-X/24 and dest_id=1;
```

Once you have retained the redo logs, you can use log miner as described in the Oracle documentation.

Previous Methods for Accessing Alert Logs and Listener Logs

You can view the alert and listener logs using the Amazon RDS console. You can also use the following methods to access these logs:

To access the alert log, use the following command:

```
select message_text from alertlog;
```

To access the listener log, use the following command:

```
select message_text from listenerlog;
```

Note
Oracle rotates the alert and listener logs when they exceed 10MB, at which point they will be unavailable from the Amazon RDS views.

SQL Server Database Log Files

You can access SQL Server error logs, agent logs, and trace files by using the Amazon RDS console or APIs. Log files are rotated each day and when a database reboot occurs; a maximum of seven files are retained for each error log, agent log, and trace file. Log files are deleted after seven days. For more information on generating logs using SQL Server Agent with Amazon RDS, see Using SQL Server Agent (p. 346)

Viewing the SQL Server Error Log Using the CLI

You can use the SQL Server specific **rds_read_error_log** CLI command to view an error log.

```
EXEC rdsadmin.dbo.rds_read_error_log [index]
```

In the **rds_read_error_log** command, *index* corresponds to the requested error log relative to the current error log. The default value is 0, which returns the current error log. The previous log has index value 1, the one before that 2, and so on.

Managing Trace Files

This section describes Amazon RDS-specific procedures to create, refresh, access, and delete trace files.

Generating a Trace SQL Query

```
declare @rc int
declare @TraceID int
declare @maxfilesize bigint
set @maxfilesize = 5
exec @rc = sp_trace_create @TraceID output,  0, N'D:\rdsdbdata\rdstest', @max
filesize, NULL
```

Viewing an Open Trace

```
select * from ::fn_trace_getinfo(default)
```

Viewing Trace Contents

```
select * from ::fn_trace_gettable('D:\rdsdbdata\rdstest.trc', default)
```

Purging Trace Files

Trace files can accumulate and consume disk space. Amazon RDS purges trace files by default and log files that are older than seven days.

To view the current trace file retention period, use the `rds_show_configuration` command. At a command prompt, type the following, and then press Enter:

```
PROMPT> exec rdsadmin..rds_show_configuration;
```

To modify the retention period for trace files, use the `rds_set_configuration` command, setting the `tracefile retention` argument to the new retention period, in minutes. The following example sets the retention period to 24 hours:

```
PROMPT> exec rdsadmin..rds_set_configuration 'tracefile retention',1440;
```

For security reasons, you cannot delete a specific trace file on a SQL Server DB instance. To delete all unused tracefiles, set the `tracefile retention` argument to 0.

PostgreSQL Database Log Files

You can set the retention period for system logs using the `rds.log_retention_period` parameter in the DB parameter group associated with your DB instance. The unit for this parameter is minutes; for example, a setting of 1440 would retain logs for one day. The default value is 4320 (three days). The maximum value is 10080 (seven days). Note that your instance must have enough allocated storage to contain the retained log files.

You can enable query logging for your PostgreSQL DB instance by setting two parameters in the DB parameter group associated with your DB instance: `log_statement` and `log_min_duration`. The `log_statement` parameter controls which SQL statements are logged. We recommend setting this parameter to *all* to log all statements; the default value is *none*. Alternatively, you can set this value to *ddl* to log all data definition statements (CREATE, ALTER, DROP, etc) or to *mod* to log all ddl and data-modifying statements (INSERT, UPDATE, DELETE, etc.).

The `log_min_duration` parameter sets the limit in milliseconds of a statement to be logged. All SQL statements that run longer than the parameter setting are logged. This parameter is disabled and set to minus 1 (-1) by default. Enabling this parameter can help you find unoptimized queries. For more information on these settings, see Error Reporting and Logging in the PostgreSQL documentation.

If you are new to setting parameters in a DB parameter group and associating that parameter group with a DB instance, see Working with DB Parameter Groups (p. 523)

The following steps show how to set up query logging:

1. Set the `log_statement` parameter to *all*. The following example shows the information that is written to the postgres.log file:

   ```
   2013-11-05 16:48:56 UTC::@:[2952]:LOG:  received SIGHUP, reloading configura
   tion files
   2013-11-05 16:48:56 UTC::@:[2952]:LOG:  parameter "log_min_duration_statement"
    changed to "1"
   ```

   Additional information is written to the postgres.log file when you execute a query. The following example shows the type of information written to the file after a query:

   ```
   2013-11-05 16:41:07 UTC::@:[2955]:LOG:  checkpoint starting: time
   ```

```
2013-11-05 16:41:07 UTC::@:[2955]:LOG:  checkpoint complete: wrote 1 buffers
 (0.3%); 0 transaction log file(s) added, 0 removed, 1 recycled; write=0.000
 s, sync=0.003 s, total=0.012 s; sync files=1, longest=0.003 s, average=0.003
 s
2013-11-05 16:45:14 UTC:[local]:master@postgres:[8839]:LOG:  statement: SELECT
 d.datname as "Name",
        pg_catalog.pg_get_userbyid(d.datdba) as "Owner",
        pg_catalog.pg_encoding_to_char(d.encoding) as "Encoding",
        d.datcollate as "Collate",
        d.datctype as "Ctype",
        pg_catalog.array_to_string(d.datacl, E'\n') AS "Access privileges"
 FROM pg_catalog.pg_database d
 ORDER BY 1;
2013-11-05 16:45:
```

2. Set the `log_min_duration` parameter. The following example shows the information that is written to the postgres.log file when the parameter is set to *1*:

```
2013-11-05 16:48:56 UTC::@:[2952]:LOG:  received SIGHUP, reloading configura
tion files
2013-11-05 16:48:56 UTC::@:[2952]:LOG:  parameter "log_min_duration_statement"
 changed to "1"
```

   Additional information is written to the postgres.log file when you execute a query that exceeds the duration parameter setting. The following example shows the type of information written to the file after a query:

```
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG:  statement: SELECT
 c2.relname, i.indisprimary, i.indisunique, i.indisclustered, i.indisvalid,
pg_catalog.pg_get_indexdef(i.indexrelid, 0, true),
   pg_catalog.pg_get_constraintdef(con.oid, true), contype, condeferrable,
condeferred, c2.reltablespace
 FROM pg_catalog.pg_class c, pg_catalog.pg_class c2, pg_catalog.pg_index i
   LEFT JOIN pg_catalog.pg_constraint con ON (conrelid = i.indrelid AND
conindid = i.indexrelid AND contype IN ('p','u','x'))
 WHERE c.oid = '1255' AND c.oid = i.indrelid AND i.indexrelid = c2.oid
 ORDER BY i.indisprimary DESC, i.indisunique DESC, c2.relname;
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG:  duration: 3.367
 ms
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG:  statement: SELECT
 c.oid::pg_catalog.regclass FROM pg_catalog.pg_class c, pg_catalog.pg_inherits
 i WHERE c.oid=i.inhparent AND i.inhrelid = '1255' ORDER BY inhseqno;
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG:  duration: 1.002
 ms
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG:  statement: SELECT
 c.oid::pg_catalog.regclass FROM pg_catalog.pg_class c, pg_catalog.pg_inherits
 i WHERE c.oid=i.inhrelid AND i.inhparent = '1255' ORDER BY c.oid::pg_cata
log.regclass::pg_catalog.text;
2013-11-05 16:51:18 UTC:[local]:master@postgres:[9193]:LOG:  statement: select
 proname from pg_proc;
2013-11-05 16:51:18 UTC:[local]:master@postgres:[9193]:LOG:  duration: 3.469
```

```
ms
```

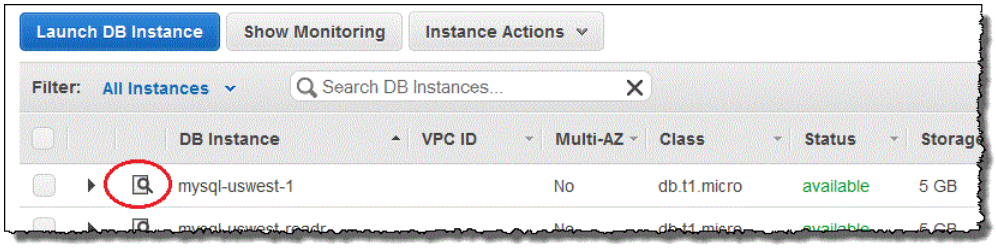
Viewing and Listing Database Log Files

You can view database log files for your DB engine by using the Amazon RDS console. You can list what log files are available for download or monitoring by using the Amazon RDS Command Line Interface (CLI) or APIs.

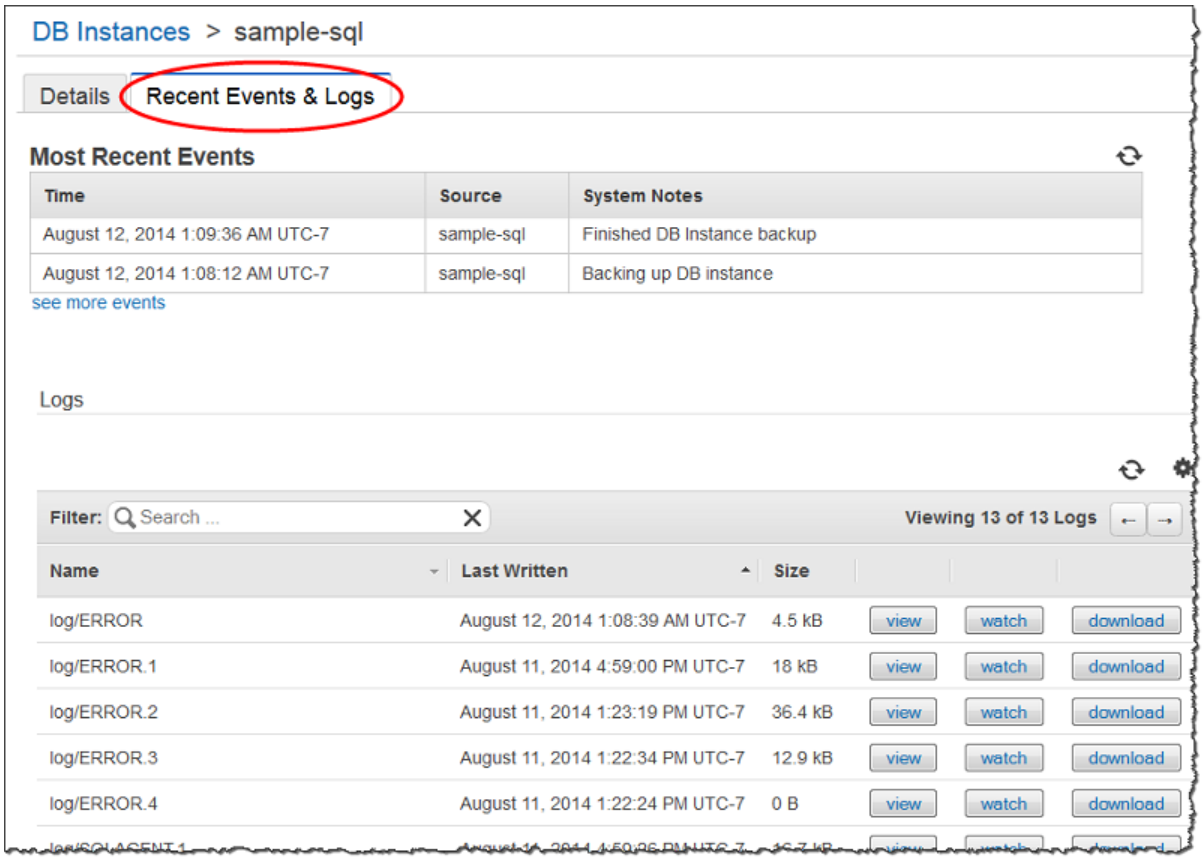
AWS Management Console

To view a database log file

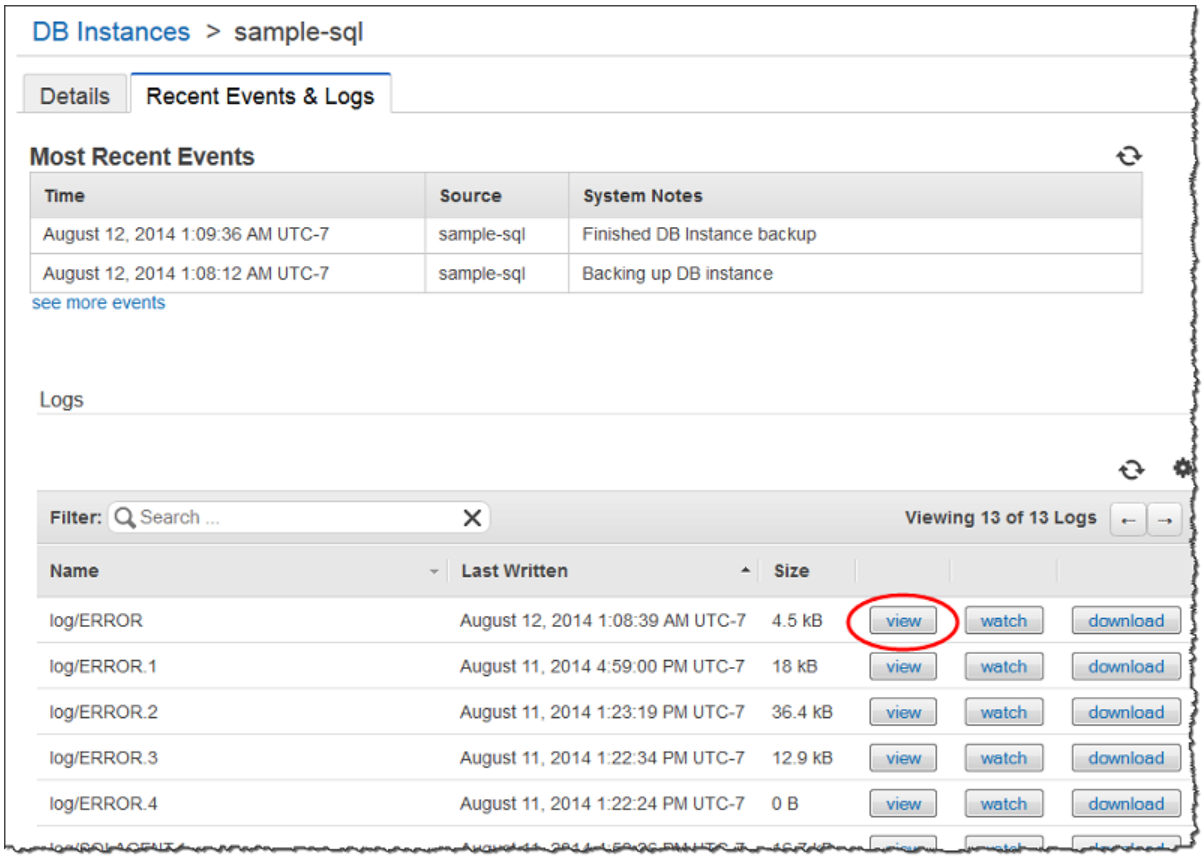
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Instances**.
3. Click the details icon next to the DB instance name that has the log file you want to view to show the DB instance details page.



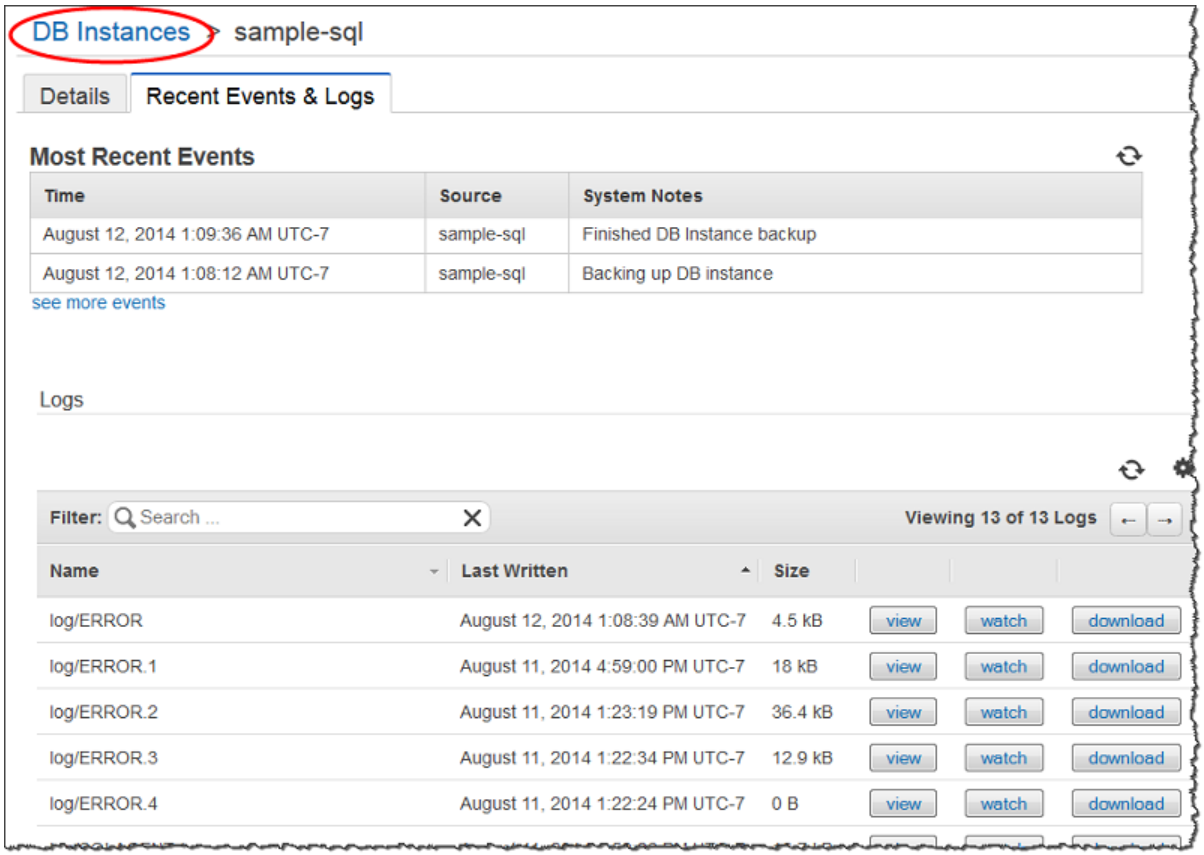
4. On the DB instance details page, click the **Recent Events & Logs** tab.



5. Click the **View** button for the log you want to view.



6. Click **DB Instances** at the top of the page to return to the list of DB instances.



CLI

To list the available database log files for a DB instance

- Use the command **rds-describe-db-log-files**.

  The following example directs a list of log files for a DB instance to a text file called log_file_list.txt.

```
PROMPT>rds-describe-db-log-files > log_file_list.txt
```

API

To list the available database log files for a DB instance

- Call `DescribeDBLogFiles`.

Related Topics

- Monitoring Amazon RDS (p. 571)
- Using Amazon RDS Event Notification (p. 575)

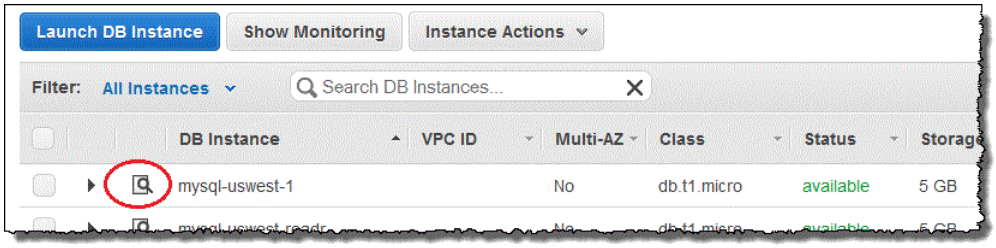
Downloading a Database Log File

You can use the Amazon RDS console or the Command Line Interface (CLI) to download a database log file. You cannot download log files using the Amazon RDS API.

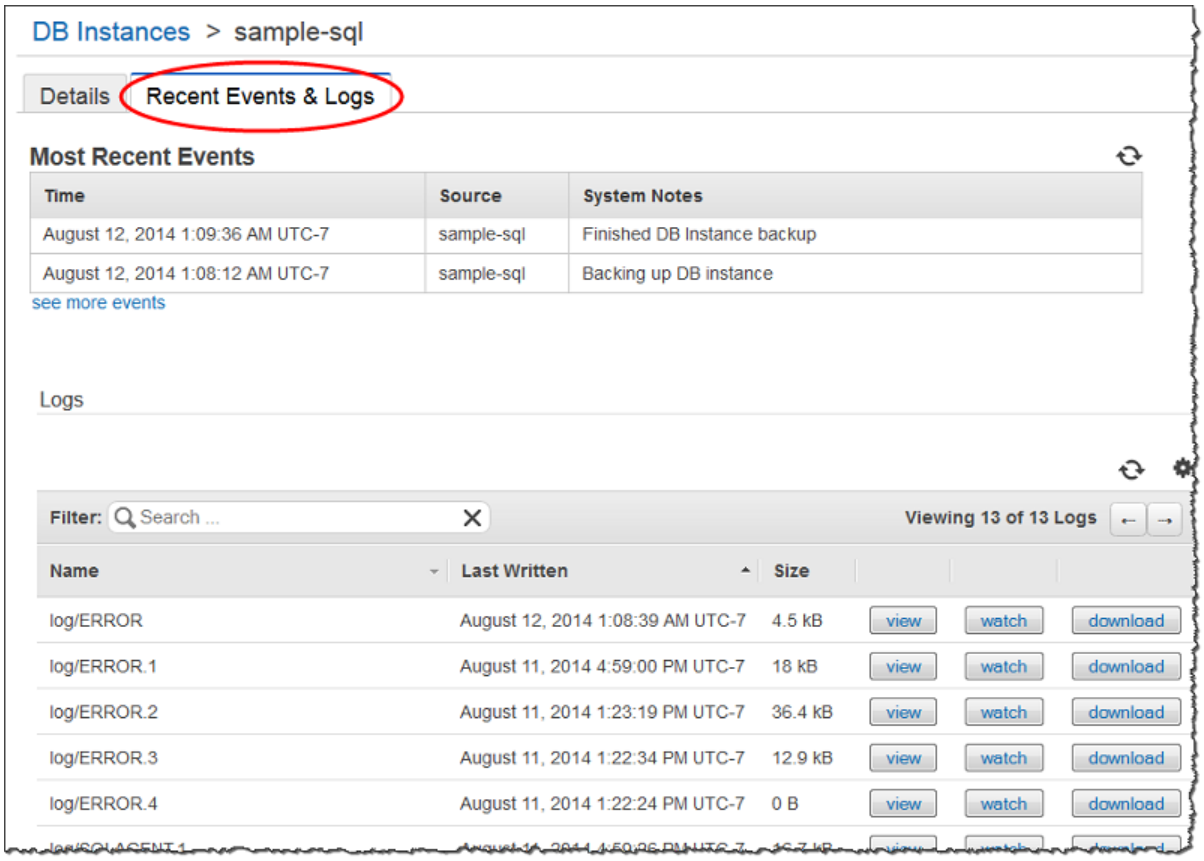
AWS Management Console

To download a database log file

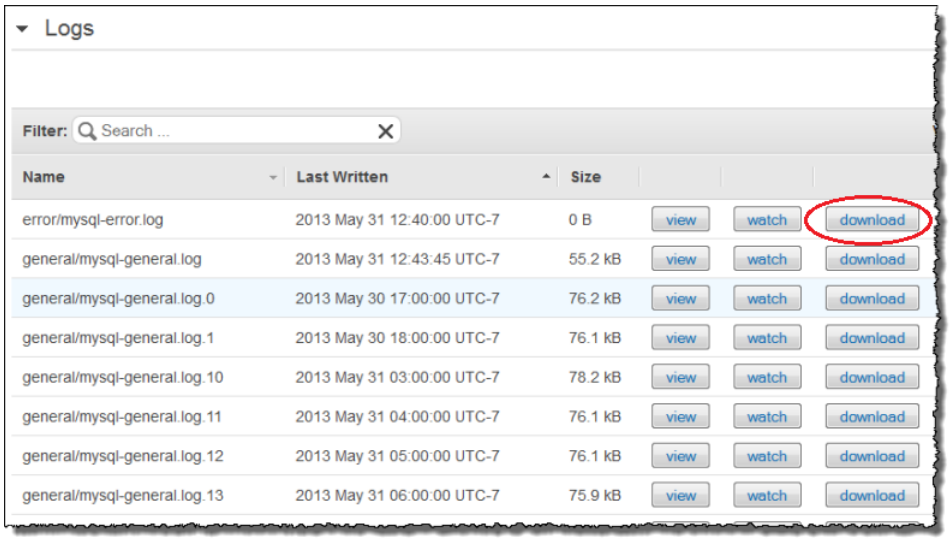
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Instances**.
3. Click the details icon for the DB instance name that has the log file you want to view.



4. On the DB instance details page, click the **Recent Events & Logs** tab.



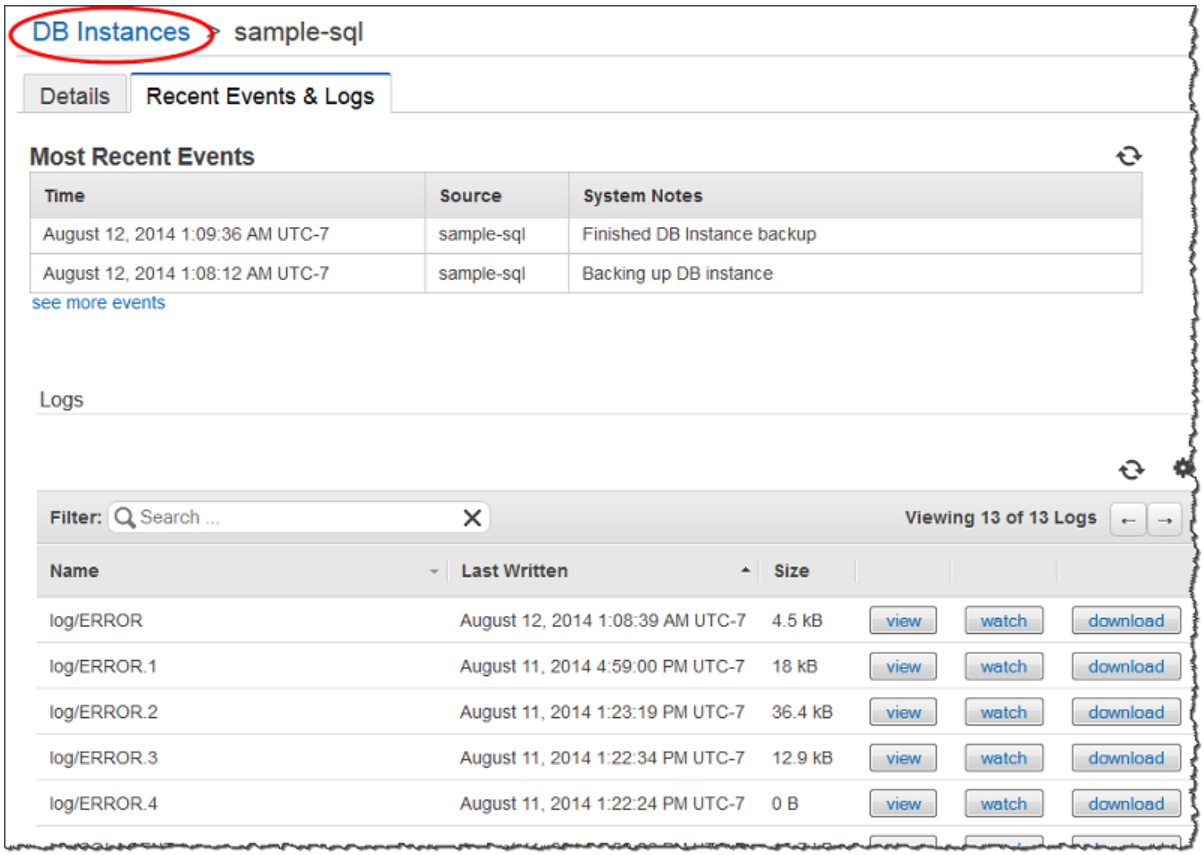
5. Click the **Download** button for the log you want to download.



6. Right-click the link provided, and then select **Save Link As...** from the dropdown menu. Type the location where you want the log file to be saved, then click **Save**. Click **Close** when you are finished.

Download Log: general/mysql-general.log.0 (76.2 kB)
Please right-click the download link below and choose "Save Link As..."
Log Name: general/mysql-general.log.0
Close

7. Click **DB Instances** at the top of the page to return to the list of DB instances.



CLI

To download a database log file

- Use the command **rds-download-db-logfile**.

  The following example shows how to download the contents of a log file called log/ERROR.4 and store it in a local file called errorlog.txt.

```
PROMPT>rds-download-db-logfile myexampledb --region us-west-2 --log-file-
name log/ERROR.4 > errorlog.txt
```

Related Topics

- Monitoring Amazon RDS (p. 571)
- Using Amazon RDS Event Notification (p. 575)

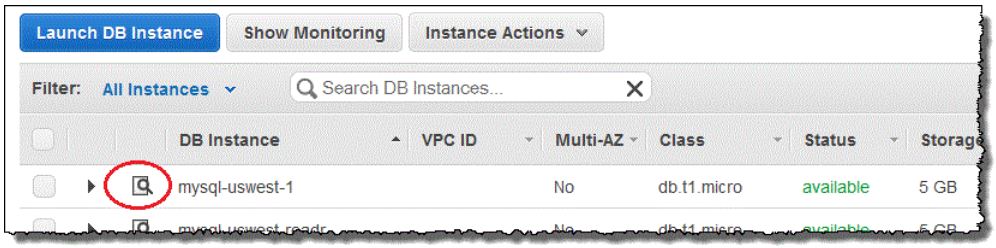
Watching a Database Log File

You can monitor the contents of a log file by using the Amazon RDS console, CLI, or API.

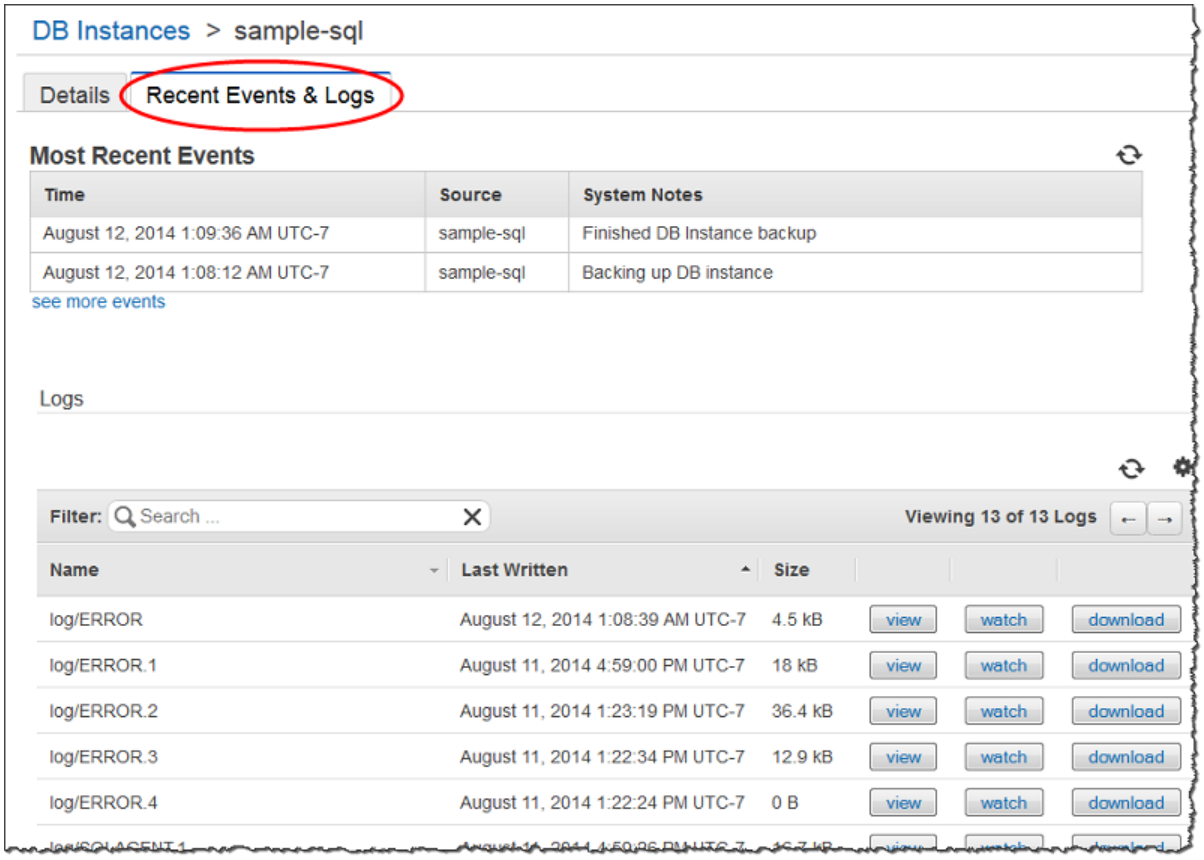
AWS Management Console

To watch a database log file

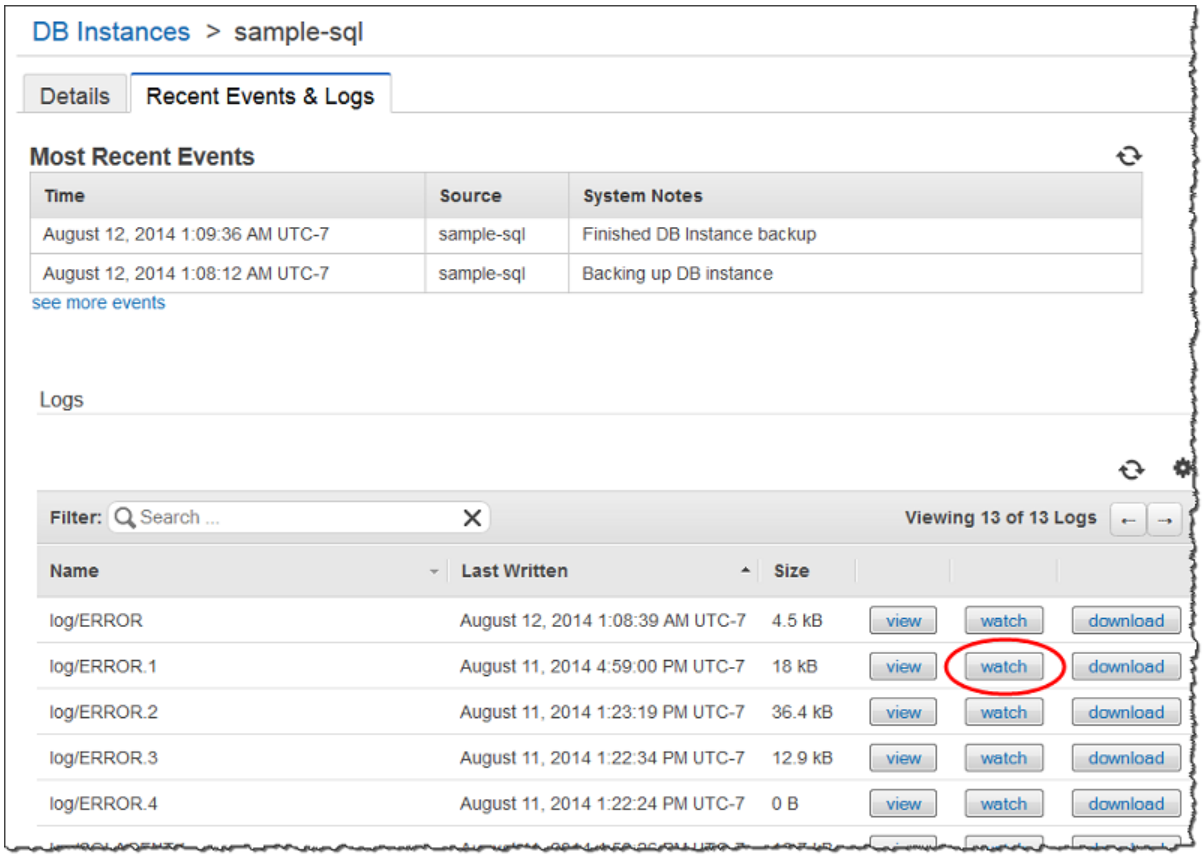
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.amazonaws.cn/rds/.
2. In the navigation pane, click **Instances**.
3. Click the details icon for the DB instance name that has the log file you want to view.



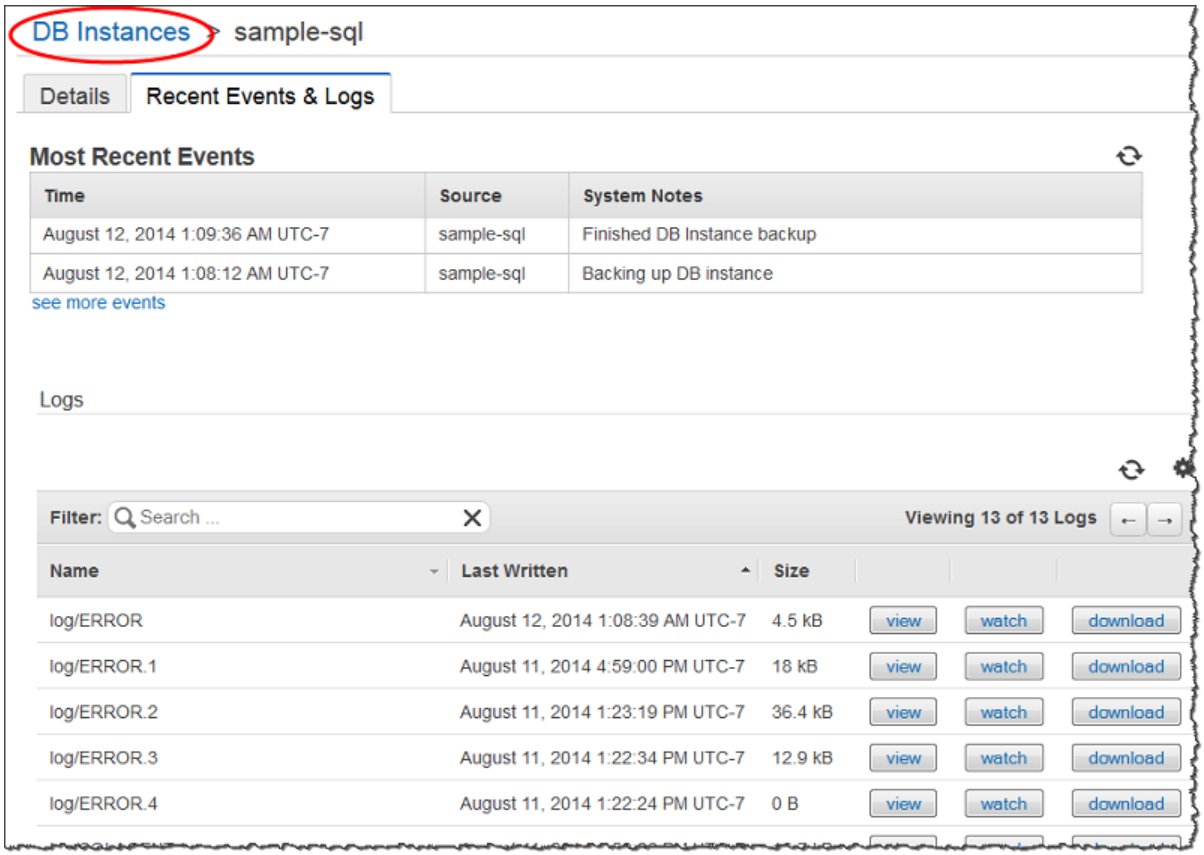
4. On the DB instance details page, click the **Recent Events & Logs** tab.



5. Click the **Watch** button for the log you want to watch.



6. Click **DB Instances** at the top of the page to return to the list of DB instances.



CLI

To watch a database log file

- Use the command **rds-watch-db-logfile**.

  The following example shows how to monitor a log file for a DB instance named mysql-db1

```
PROMPT>rds-watch-db-logfile mysql-db1 --log-file-name error-running.log.20
```

API

To watch a database log file

- Call `DownloadDBLogFilePortion`.

Related Topics

- Monitoring Amazon RDS (p. 571)
- Using Amazon RDS Event Notification (p. 575)

Logging Amazon RDS API Calls Using AWS CloudTrail

AWS CloudTrail is a service that logs all Amazon RDS API calls made by or on behalf of your AWS account. The logging information is stored in an Amazon S3 bucket. You can use the information collected by CloudTrail to monitor activity for your Amazon RDS DB instances. For example, you can determine whether a request completed successfully and which user made the request. To learn more about CloudTrail, see the AWS CloudTrail User Guide.

If an action is taken on behalf of your AWS account using the Amazon RDS console or the Amazon RDS command line interface, then AWS CloudTrail will log the action as calls made to the Amazon RDS API. For example, if you use the Amazon RDS console to modify a DB instance, or call the rds-modify-db-instance CLI command, then the AWS CloudTrail log will show a call to the ModifyDBInstance API action. For a list of the Amazon RDS API actions that are logged by AWS CloudTrail, go to Amazon RDS API Reference.

Note
AWS CloudTrail only logs events for Amazon RDS API calls. If you want to audit actions taken on your database that are not part of the Amazon RDS API, such as when a user connects to your database or when a change is made to your database schema, then you will need to use the monitoring capabilities provided by your DB engine.

Configuring CloudTrail Event Logging

CloudTrail creates audit trails in each region separately and stores them in an Amazon S3 bucket. You can configure CloudTrail to use Amazon SNS to notify you when a log file is created, but that is optional. CloudTrail will notify you frequently, so we recommend that you use Amazon SNS in conjunction with an Amazon SQS queue and handle notifications programmatically.

You can enable CloudTrail using the AWS Management Console, CLI, or API. When you enable CloudTrail logging, you can have the CloudTrail service create an Amazon S3 bucket for you to store your log files. For details, see Creating and Updating Your Trail in the *AWS CloudTrail User Guide*. The *AWS CloudTrail User Guide* also contains information on how to aggregate CloudTrail logs from multiple regions into a single Amazon S3 bucket.

There is no cost to use the CloudTrail service. However, standard rates for Amazon S3 usage apply as well as rates for Amazon SNS usage should you include that option. For pricing details, see the Amazon S3 and Amazon SNS pricing pages.

Amazon RDS Event Entries in CloudTrail Log Files

CloudTrail log files contain event information formatted using JSON. An event record represents a single AWS API call and includes information about the requested action, the user that requested the action, the date and time of the request, and so on.

CloudTrail log files include events for all AWS API calls for your AWS account, not just calls to the Amazon RDS API. However, you can read the log files and scan for calls to the Amazon RDS API using the `eventName` element.

The following example shows a CloudTrail log for a user that created a snapshot of a DB instance and then deleted that instance using the Amazon RDS console. The console is identified by the `userAgent` element. The requested API calls made by the console (`CreateDBSnapshot` and `DeleteDBInstance`) are found in the `eventName` element for each record. Information about the user (`Alice`) can be found in the `userIdentity` element.

```
{
  Records:[
  {
    "awsRegion":"us-west-2",
    "eventName":"CreateDBSnapshot",
    "eventSource":"rds.amazonaws.com",
    "eventTime":"2014-01-14T16:23:49Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-ka
os_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"AKIADQKE4SARGYLE",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  },
  {
    "awsRegion":"us-west-2",
    "eventName":"DeleteDBInstance",
    "eventSource":"rds.amazonaws.com",
    "eventTime":"2014-01-14T16:28:27Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-ka
os_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"AKIADQKE4SARGYLE",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  }
  ]
```

```
}
```

For more information about the different elements and values in CloudTrail log files, see CloudTrail Event Reference in the *AWS CloudTrail User Guide*.

You may also want to make use of one of the Amazon partner solutions that integrate with CloudTrail to read and analyze your CloudTrail log files. For options, see the AWS partners page.

Troubleshooting

The following sections can help you troubleshoot problems you have with Amazon RDS.

Topics

- Cannot Connect to Amazon RDS DB Instance (p. 618)
- Amazon RDS Security Issues (p. 619)
- Amazon RDS DB Instance Outage or Reboot (p. 620)
- Amazon RDS DB Parameter Changes Not Taking Effect (p. 620)
- Amazon RDS DB Instance Running Out of Storage (p. 621)
- Amazon RDS MySQL Issues (p. 622)
- Amazon RDS Oracle GoldenGate Issues (p. 628)
- Cannot Connect to Amazon RDS SQL Server DB Instance (p. 628)
- Cannot Connect to Amazon RDS PostgreSQL DB Instance (p. 629)

Cannot Connect to Amazon RDS DB Instance

When you cannot connect to a DB instance, the following are common causes:

- The access rules enforced by your local firewall and the ingress IP addresses that you authorized to access your DB instance in the instance's security group are not in sync. The problem is most likely the ingress rules in your security group. By default, DB instances do not allow access; access is granted through a security group. To grant access, you must create your own security group with specific ingress and egress rules for your situation. For more information about setting up a security group, see Create a Security Group (p. 10).
- The port you specified when you created the DB instance cannot be used to send or receive communications due to your local firewall restrictions. In this case, check with your network administrator to determine if your network allows the specified port to be used for inbound and outbound communication.
- Your DB instance is still being created and is not yet available. Depending on the size of your DB instance, it can take up to 20 minutes before an instance is available.

Testing a Connection to an Amazon RDS DB Instance

You can test your connection to a DB instance using common Linux or Windows tools.

From a Linux or Unix terminal, you can test the connection by typing the following (replace *<DB-instance-endpoint>* with the endpoint and *<port>* with the port of your DB instance):

```
$nc -zv <DB-instance-endpoint> <port>
```

For example, the following shows a sample command and the return value:

```
$nc -zv postgresql1.c6c8mn7tsdgv0.us-west-2.rds.amazonaws.com 8299

Connection to postgresql1.c6c8mn7tsdgv0.us-west-2.rds.amazonaws.com 8299 port
[tcp/vvr-data] succeeded!
```

Windows users can use Telnet to test the connection to a DB instance. Note that Telnet actions are not supported other than for testing the connection. If a connection is successful, the action returns no message. If a connection is not successful, you receive an error message such as the following:

```
C:\>telnet sg-postgresql1.c6c8mntzhgv0.us-west-2.rds.amazonaws.com 819

Connecting To sg-postgresql1.c6c8mntzhgv0.us-west-2.rds.amazonaws.com...Could
not open
connection to the host, on port 819: Connect failed
```

If Telnet actions return success, your security group is properly configured.

Troubleshooting Connection Authentication

If you can connect to your DB instance but you get authentication errors, you might want to reset the master user password for the DB instance. You can do this by modifying the RDS instance; for more information, see one of the following topics:

- Modifying a DB Instance Running the MySQL Database Engine (p. 140)
- Modifying a DB Instance Running the Oracle Database Engine (p. 216)
- Modifying a DB Instance Running the SQL Server Database Engine (p. 328)
- Modifying a DB Instance Running the PostgreSQL Database Engine (p. 374)

Amazon RDS Security Issues

To avoid security issues, never use your master AWS user name and password for a user account. Best practice is to use your master AWS account to create IAM users and assign those to DB user accounts.

You can also use your master account to create other user accounts, if necessary. For more information on creating IAM users, see Create an IAM User (p. 8).

Amazon RDS DB Instance Outage or Reboot

A DB instance outage can occur when a DB instance is rebooted, when the DB instance is put into a state that prevents access to it, and when the database is restarted. A reboot can occur when you manually reboot your DB instance or when you change a DB instance setting that requires a reboot before it can take effect.

When you modify a setting for a DB instance, you can determine when the change is applied by using the **Apply Immediately** setting. To see a table that shows DB instance actions and the effect that setting the **Apply Immediately** value has, see Modifying a DB Instance and Using the Apply Immediately Parameter (p. 455).

A DB instance reboot only occurs when you change a setting that requires a reboot, or when you manually cause a reboot. A reboot can occur immediately if you change a setting and request that the change take effect immediately or it can occur during the DB instance's maintenance window.

A DB instance reboot occurs immediately when one of the following occurs:

- You change the backup retention period for a DB instance from 0 to a nonzero value or from a nonzero value to 0 and set **Apply Immediately** to *true*.
- You change the DB instance class, and **Apply Immediately** is set to *true*.
- You change the storage type from **Magnetic (Standard)** to **General Purpose (SSD**) or **Provisioned IOPS (SSD)**, or from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic (Standard)**. from standard to PIOPS.

A DB instance reboot occurs during the maintenance window when one of the following occurs:

- You change the backup retention period for a DB instance from 0 to a nonzero value or from a nonzero value to 0, and **Apply Immediately** is set to *false*.
- You change the DB instance class, and **Apply Immediately** is set to *false*.

When you change a static parameter in a DB parameter group, the change will not take effect until the DB instance associated with the parameter group is rebooted. The change requires a manual reboot; the DB instance will not automatically be rebooted during the maintenance window.

Amazon RDS DB Parameter Changes Not Taking Effect

If you change a parameter in a DB parameter group but you don't see the changes take effect, you most likely need to reboot the DB instance associated with the DB parameter group. When you change a dynamic parameter, the change takes effect immediately; when you change a static parameter, the change won't take effect until you reboot the DB instance associated with the parameter group.

You can reboot a DB instance using the RDS console or explicitly calling the `RebootDbInstance` API action (without failover, if the DB instance is in a Multi-AZ deployment). The requirement to reboot the associated DB instance after a static parameter change helps mitigate the risk of a parameter misconfiguration affecting an API call, such as calling `ModifyDBInstance` to change DB instance class or scale storage. For more information, see Modifying Parameters in a DB Parameter Group (p. 525).

Amazon RDS DB Instance Running Out of Storage

If your DB instance runs out of storage space, it might no longer be available. We highly recommend that you constantly monitor the `FreeStorageSpace` metric published in CloudWatch to ensure that your DB instance has enough free storage space.

If your database instance runs out of storage, its status will change to *storage-full*. For example, a call to the `DescribeDBInstances` action for a DB instance that has used up its storage will output the following:

```
PROMPT> rds-describe-db-instances mydbinstance
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m1.large  mysql5.1  50
 sa
storage-full  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3
 SECGROUP  default  active
 PARAMGRP  default.mysql5.1  in-sync
```

To recover from this scenario, add more storage space to your instance using the `ModifyDBInstance` action or the following CLI command:

```
PROMPT>rds-modify-db-instance mydbinstance --allocated-storage 60 --apply-imme
diately
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m1.large  mysql5.1  50
 sa
storage-full  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3  60
 SECGROUP  default  active
 PARAMGRP  default.mysql5.1  in-sync
```

Now, when you describe your DB instance, you will see that your DB instance will have *modifying* status, which indicates the storage is being scaled.

```
PROMPT>rds-describe-db-instances mydbinstance
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m1.large  mysql5.1  50
 sa
modifying  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com
3306  us-east-1b  3  60
 SECGROUP  default  active
 PARAMGRP  default.mysql5.1  in-sync
```

Once storage scaling is complete, your DB instance status will change to *available*.

```
PROMPT>rds-describe-db-instances mydbinstance
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m1.large  mysql5.1  60
 sa
available  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3
 SECGROUP  default  active
 PARAMGRP  default.mysql5.1  in-sync
```

Note that you can receive notifications when your storage space is exhausted using the `DescribeEvents` action. For example, in this scenario, if you do a `DescribeEvents` call after these operations you will see the following output:

```
PROMPT>rds-describe-events --source-type db-instance --source-identifier
mydbinstance
2009-12-22T23:44:14.374Z  mydbinstance  Allocated storage has been exhausted
db-instance
2009-12-23T00:14:02.737Z  mydbinstance  Applying modification to allocated
storage db-instance
2009-12-23T00:31:54.764Z  mydbinstance  Finished applying modification to alloc
ated storage
```

Amazon RDS MySQL Issues

MySQL Version 5.5.40 Asynchronous I/O Is Disabled

You might observe reduced I/O performance if you have a MySQL DB instance that was created before April 23, 2014 and then upgraded to MySQL version 5.5.40 after October 17, 2014. This reduced performance can be caused by an error that disables the `innodb_use_native_aio` parameter even if the corresponding DB parameter group enables the `innodb_use_native_aio` parameter.

To resolve this error, we recommend that you upgrade your MySQL DB instance running version 5.5.40 to version 5.5.40a, which corrects this behavior. For information on minor version upgrades, see Minor DB Engine Version Upgrades for a DB Instance (p. 447).

For more information on MySQL asynchronous I/O, go to Asynchronous I/O on Linux in the MySQL documentation.

Index Merge Optimization Returns Wrong Results

Queries that use index merge optimization might return wrong results due to a bug in the MySQL query optimizer that was introduced in MySQL 5.5.37. When you issue a query against a table with multiple indexes the optimizer scans ranges of rows based on the multiple indexes, but does not merge the results together correctly. For more information on the query optimizer bug, go to http://bugs.mysql.com/bug.php?id=72745 and http://bugs.mysql.com/bug.php?id=68194 in the MySQL bug database.

For example, consider a query on a table with two indexes where the search arguments reference the indexed columns.

```
SELECT * FROM table1
  WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

In this case, the search engine will search both indexes. However, due to the bug, the merged results will be incorrect.

To resolve this issue, you can do one of the following:

- Set the `optimizer_switch` parameter to `index_merge=off` in the DB parameter group for your MySQL DB instance. For information on setting DB parameter group parameters, see Working with DB Parameter Groups (p. 523).
- Upgrade your MySQL DB instance to MySQL version 5.6.19a. For information on major version upgrades, see Major DB Engine Version Upgrades for a DB Instance (p. 447).
- If you cannot upgrade your instance or change the `optimizer_switch` parameter, you can work around the bug by explicitly identifying an index for the query, for example:

```
SELECT * FROM table1
  USE INDEX covering_index
  WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

For more information, go to Index Merge Optimization.

Replication Fails After Upgrading to MySQL Version 5.6.21

If you have a DB instance that runs a version prior to version 5.6.4, or if the DB instance was upgraded from a version prior to version 5.6.4, you can receive the following error if you have a Read Replica that runs MySQL version 5.6.21.

```
mysqld got signal 11 ;
This could be because you hit a bug. It is also possible that this binary
or one of the libraries it was linked against is corrupt, improperly built,
or misconfigured. This error can also be caused by malfunctioning hardware.
We will try our best to scrape up some info that will hopefully help
diagnose the problem, but since we have already crashed,
something is definitely wrong and this may fail.
```

MySQL version 5.6.4 introduced a new date and time format for `datetime`, `time`, and `timestamp` columns that allows fractional components in date and time values. The error is caused by a mismatch in date and time formats between the master and the replica, and results in a failure when row-based logging attempts to replay an operation from the master DB instance to the replica DB instance. You might also see a number of related row-based logging messages in your MySQL error log, for example: `Relay_log_info`, `Rows_log_event`, and so on. For information on the new date and time format for MySQL, go to Upgrading from MySQL 5.5 to 5.6 in the MySQL documentation.

To resolve the error, you can do either of the following:

- Upgrade your Read Replica to MySQL version 5.6.23 or later. For information on upgrading a MySQL DB instance on Amazon RDS to version 5.6, see Upgrading Database Versions for a DB Instance (p. 446).
- Upgrade your master DB instance to MySQL version 5.6.12 or later and update the format of the affected date and time columns. For information on upgrading a MySQL DB instance on Amazon RDS to version 5.6, see Upgrading Database Versions for a DB Instance (p. 446).

  To upgrade your date and time columns to the new format on your master DB instance, you must issue the `ALTER TABLE <table_name> FORCE;` command.

Note
Because altering a table locks the table as read-only, we recommend that you perform this update during a maintenance window.

You can run the following query to find all of the tables in your database that have columns of type `datetime`, `time`, or `timestamp` and create an `ALTER TABLE <table_name> FORCE;` command for each table.

```
SELECT DISTINCT CONCAT('ALTER TABLE `',
  REPLACE(is_tables.TABLE_SCHEMA, '`', '``'), '`.`',
  REPLACE(is_tables.TABLE_NAME, '`', '``'), '` FORCE;')
FROM information_schema.TABLES is_tables
  INNER JOIN information_schema.COLUMNS col ON col.TABLE_SCHEMA =
is_tables.TABLE_SCHEMA
    AND col.TABLE_NAME = is_tables.TABLE_NAME
  LEFT OUTER JOIN information_schema.INNODB_SYS_TABLES systables ON
    SUBSTRING_INDEX(systables.NAME, '#', 1) = CON
CAT(is_tables.TABLE_SCHEMA,'/',is_tables.TABLE_NAME)
  LEFT OUTER JOIN information_schema.INNODB_SYS_COLUMNS syscolumns ON
    syscolumns.TABLE_ID = systables.TABLE_ID AND syscolumns.NAME =
col.COLUMN_NAME
WHERE col.COLUMN_TYPE IN ('time','timestamp','datetime')
  AND is_tables.TABLE_TYPE = 'BASE TABLE'
  AND is_tables.TABLE_SCHEMA NOT IN ('mysql','information_schema','perform
ance_schema')
  AND (is_tables.ENGINE = 'InnoDB' AND syscolumns.MTYPE = 6);
```

Diagnosing and Resolving Lag Between Read Replicas

After you create a MySQL Read Replica and the Read Replica is available, Amazon RDS first replicates the changes made to the source DB instance from the time the create Read Replica operation was initiated. During this phase, the replication lag time for the Read Replica will be greater than 0. You can monitor this lag time in Amazon CloudWatch by viewing the Amazon RDS `ReplicaLag` metric.

The `ReplicaLag` metric reports the value of the `Seconds_Behind_Master` field of the MySQL `SHOW SLAVE STATUS` command. For more information, see SHOW SLAVE STATUS. When the `ReplicaLag` metric reaches 0, the replica has caught up to the source DB instance. If the `ReplicaLag` metric returns -1, replication might not be active. To troubleshoot a replication error, see Diagnosing and Resolving a MySQL Read Replication Failure (p. 625). A `ReplicaLag` value of -1 can also mean that the `Seconds_Behind_Master` value cannot be determined or is `NULL`.

The `ReplicaLag` metric will return -1 during a network outage or when a patch is applied during the maintenance window. In this case, wait for network connectivity to be restored or for the maintenance window to end before you check the `ReplicaLag` metric again.

Because the MySQL read replication technology is asynchronous, you can expect occasional increases for the `BinLogDiskUsage` metric on the source DB instance and for the `ReplicaLag` metric on the Read Replica. For example, a high volume of write operations to the source DB instance can occur in parallel, while write operations to the Read Replica are serialized using a single I/O thread—this can lead to a lag between the source instance and Read Replica. For more information about Read Replicas and MySQL, go to Replication Implementation Details in the MySQL documentation.

You can reduce the lag between updates to a source DB instance and the subsequent updates to the Read Replica by doing the following:

- Set the DB instance class of the Read Replica to have a storage size comparable to that of the source DB instance.
- Ensure that parameter settings in the DB parameter groups used by the source DB instance and the Read Replica are compatible. For more information and an example, see the discussion of the `max_allowed_packet` parameter in the next section.
- Disable the query cache. For tables that are modified often, using the query cache can increase replica lag because the cache is locked and refreshed often. If this is the case, you might see less replica lag if you disable the query cache. You can disable the query cache by setting the `query_cache_type parameter` to 0 in the DB parameter group for the DB instance. For more information on the query cache, see Query Cache Configuration.
- Warm the InnoDB buffer pool on the Read Replica. If you have a small set of tables that are being updated often, and you are using the InnoDB table schema, then dump those tables on the Read Replica. Doing this will cause MySQL to scan through the rows of those tables from the disk and then cache them in the buffer pool, which can reduce replica lag. For example:

```
PROMPT> mysqldump –h <endpoint> --port=<port> -u=<username> -p <password>
database_name table1 table2 > /dev/null
```

Diagnosing and Resolving a MySQL Read Replication Failure

Amazon RDS monitors the replication status of your Read Replicas and updates the **Replication State** field of the Read Replica instance to **Error** if replication stops for any reason. You can review the details of the associated error thrown by the MySQL engine by viewing the **Replication Error** field. Events that indicate the status of the Read Replica are also generated, including RDS-EVENT-0045 (p. 579), RDS-EVENT-0046 (p. 579), and RDS-EVENT-0047 (p. 578). For more information about events and subscribing to events, see Using Amazon RDS Event Notification (p. 575). If a MySQL error message is returned, review the error in the MySQL error message documentation.

Common situations that can cause replication errors include:

- The value for the `max_allowed_packet` parameter for a Read Replica is less than the `max_allowed_packet` parameter for the source DB instance.

  The `max_allowed_packet` parameter is a custom parameter that you can set in a DB parameter group that is used to specify the maximum size of data manipulation language (DML) that can be executed on the database. If the `max_allowed_packet` parameter value for the source DB instance is smaller than the `max_allowed_packet` parameter value for the Read Replica, the replication process can throw an error and stop replication. The most common error is `packet bigger than 'max_allowed_packet' bytes`. You can fix the error by having the source and Read Replica use DB parameter groups with the same `max_allowed_packet` parameter values.
- Writing to tables on a Read Replica. If you are creating indexes on a Read Replica, you need to have the `read_only` parameter set to *0* to create the indexes. If you are writing to tables on the Read Replica, it can break replication.
- Using a nontransactional storage engine such as MyISAM. Read replicas require a transactional storage engine. Replication is only supported for the InnoDB storage engine.

  You can convert a MyISAM table to InnoDB with the following command:

```
alter table <schema>.<table_name> engine=innodb;
```

- Using unsafe non-deterministic queries such as `SYSDATE()`. For more information, see Determination of Safe and Unsafe Statements in Binary Logging.

The following steps can help resolve your replication error:

- If you encounter a logical error and you can safely skip the error, follow the steps described in Skipping the Current Replication Error (p. 168). Your MySQL DB instance must be running a version that includes the `mysql_rds_skip_repl_error` procedure. For more information, see mysql_rds_skip_repl_error (p. 182).
- If you encounter a binlog position issue, you can change the slave replay position with the `mysql_rds_next_master_log` command. Your MySQL DB instance must be running a version that supports the `mysql_rds_next_master_log` command in order to change the slave replay position. For version information, see mysql.rds_next_master_log (p. 182).
- If you encounter a temporary performance issue due to high DML load, you can set the `innodb_flush_log_at_trx_commit` parameter to 2 in the DB parameter group on the Read Replica. Doing this can help the Read Replica catch up, though it temporarily reduces ACID durability.
- You can delete the Read Replica and create an instance using the same DB instance identifier so that the endpoint remains the same as that of your old Read Replica.

If a replication error is fixed, the **Replication State** changes to **replicating**. For more information, see Troubleshooting a MySQL Read Replica Problem (p. 483).

Creating Triggers with Binary Logging Enabled Requires SUPER Privilege

When trying to create triggers in an RDS MySQL DB instance, you might receive the following error:

```
"You do not have the SUPER privilege and binary logging is enabled"
```

To use triggers when binary logging is enabled requires the MySQL SUPER privilege, which is restricted for RDS MySQL DB instances. You can create triggers when binary logging is enabled without the SUPER privilege by setting the `log_bin_trust_function_creators` parameter to true. To set the `log_bin_trust_function_creators` to true, create a new DB parameter group or modify an existing DB parameter group.

To create a new DB parameter group that allows you to create triggers in your RDS MySQL DB instance with binary logging enabled, use the following CLI commands. To modify an existing parameter group, start with step 2.

To create a new parameter group to allow triggers with binary logging enabled using the CLI

1. Create a new parameter group.

   ```
   rds-create-db-parameter-group allow-triggers -d "parameter group allowing
   triggers" -f mysql5.6
   ```

2. Modify the DB parameter group to allow triggers.

```
rds-modify-db-parameter-group allow-triggers -p "name=log_bin_trust_func
tion_creators,value=true, method=pending-reboot"
```

3. Modify your DB instance to use the new DB parameter group.

```
rds-modify-db-instance <db-instance-identifier> -g allow-triggers --apply-
immediately
```

4. In order for the changes to take effect, manually reboot the DB instance.

```
rds-reboot-db-instance <db-instance-identifier>
```

Diagnosing and Resolving Point in Time Restore Failures

Restoring a DB Instance That Includes Temporary Tables

When attempting a point in time restore (PITR) of your MySQL DB instance, you might encounter the following error:

```
   Database instance could not be restored because there has been incompatible
 database activity for restore
   functionality. Common examples of incompatible activity include using tem
porary tables, in-memory tables,
   or using MyISAM tables. In this case, use of Temporary table was detected.
```

Point in time restore relies on both MySQL backup snapshots and MySQL binlogs to restore your DB instance to a particular time. Temporary table information can be unreliable in binlogs and can cause a PITR failure. If you use temporary tables in your MySQL DB instance, you can minimize the possibility of a PITR failure by performing more frequent backups. A PITR failure is most probable in the time between a temporary table's creation and the next backup snapshot.

Restoring a DB Instance That Includes In-Memory Tables

You might encounter a problem when restoring a database that has in-memory tables. In-memory tables are purged during a restart. As a result, your in-memory tables might be empty after a reboot. We recommend that when you use in-memory tables, you architect your solution to handle empty tables in the event of a restart. If you are using in-memory tables with replicated DB instances, you might need to recreate the Read Replicas after a restart if a Read Replica reboots and is unable to restore data from an empty in-memory table.

For more information about backups and PITR, see DB Instance Backups (p. 73) and Restoring a DB Instance to a Specified Time (p. 508).

Amazon RDS Oracle GoldenGate Issues

Using Oracle GoldenGate with Amazon EC2 Instances

If you are using Oracle GoldenGate with an EC2 instance, the EC2 instance must have a full installation of Oracle DBMS 11g version 11.2.0.3 and Oracle GoldenGate 11.2.1 and have Oracle patch 13328193 installed. If you do not have these items correctly installed, you will receive this error message:

```
  2014-03-06 07:09:21  ERROR   OGG-02021  This database lacks the required
libraries to support integrated capture.
```

To determine what patches you currently have installed, run the command `opatch lsinventory` on your EC2 instance.

Retaining Logs for Sufficient Time

The source database must retain archived redo logs. The duration for log retention is specified in hours. The duration should exceed any potential downtime of the source instance or any potential period of communication or networking issues for the source instance, so that Oracle GoldenGate can recover logs from the source instance as needed. The absolute minimum value required is one (1) hour of logs retained. If you don't have log retention enabled, or if the retention value is too small, you will receive the following message:

```
  2014-03-06 06:17:27  ERROR   OGG-00446  error 2 (No such file or directory)

  opening redo log /rdsdbdata/db/GGTEST3_A/onlinelog/o1_mf_2_9k4bp1n6_.log
  for sequence 1306Not able to establish initial position for begin time 2014-
03-06 06:16:55.
```

Cannot Connect to Amazon RDS SQL Server DB Instance

When you have problems connecting to a DB instance using SQL Server Management Studio, the following are some common causes:

- The access rules enforced by your local firewall and the IP addresses you authorized to access your DB instance in the instance's security group are not in sync. If you use your DB instance's endpoint and port with Microsoft SQL Server Management Studio and cannot connect, the problem is most likely the egress or ingress rules on your firewall. To grant access, you must create your own security group with specific ingress and egress rules for your situation. For more information about security groups, see Amazon RDS Security Groups (p. 110).
- The port you specified when you created the DB instance cannot be used to send or receive communications due to your local firewall restrictions. In this case, check with your network administrator

to determine if your network allows the specified port to be used for inbound and outbound communication.

- Your DB instance is still being created and is not yet available. Depending on the size of your DB instance, it can take up to 20 minutes before an instance is available.

If you can send and receive communications through the port you specified, check for the following SQL Server errors:

- **Could not open a connection to SQL Server - Microsoft SQL Server, Error: 53** – You must include the port number when you specify the server name when using Microsoft SQL Server Management Studio. For example, the server name for a DB instance (including the port number) might be: **`sqlsvr-pdz.c6c8mdfntzgv0.region.rds.amazonaws.com,1433`**.
- **No connection could be made because the target machine actively refused it - Microsoft SQL Server, Error: 10061** – In this case, you reached the DB instance but the connection was refused. This error is often caused by an incorrect user name or password.

Cannot Connect to Amazon RDS PostgreSQL DB Instance

The most common problem when attempting to connect to a PostgreSQL DB instance is that the security group assigned to the DB instance has incorrect access rules. By default, DB instances do not allow access; access is granted through a security group. To grant access, you must create your own security group with specific ingress and egress rules for your situation. For more information about creating a security group for your DB instance, see Create a Security Group (p. 10).

The most common error is `could not connect to server: Connection timed out`. If you receive this error, check that the host name is the DB instance endpoint and that the port number is correct. Check that the security group assigned to the DB instance has the necessary rules to allow access through your local firewall.

Amazon RDS API

Topics

- Using the Query API (p. 630)
- Using the SOAP API (p. 633)
- Available Libraries (p. 636)
- Troubleshooting Applications on Amazon RDS (p. 636)
- RDS REST API Reference (p. 637)

Using the Query API

The following sections discuss the parameters and request authentication used with the Query API.

Query Parameters

HTTP Query-based requests are HTTP requests that use the HTTP verb GET or POST and a Query parameter named `Action`.

Each Query request must include some common parameters to handle authentication and selection of an action.

Some operations take lists of parameters. These lists are specified using the *param.n* notation. Values of *n* are integers starting from 1.

For information about Amazon RDS regions and endpoints, go to Amazon Relational Database Service (RDS) in the Regions and Endpoints section of the *Amazon Web Services General Reference.*

Query Request Authentication

You can only send Query requests over HTTPS, and you must include a signature in every Query request. You must use either a signature version 2 or signature version 4. This section describes how to create a signature version 2. For information about creating a signature version 4, see Signature Version 4 Signing Process.

The following are the basic steps used to authenticate requests to AWS. This process assumes you are registered with AWS and have an access key ID and secret access key.

Tip
You can find your access key ID and secret access key in the Security Credentials section of the AWS Your Account page.

1. The sender constructs a request to AWS.
2. The sender calculates the request signature, a Keyed-Hashing for Message Authentication Code (HMAC) with a SHA-1 hash function, as defined in the next section of this topic.
3. The sender of the request sends the request data, the signature, and access key ID (the key identifier of the secret access key used) to AWS.
4. AWS uses the access key ID to look up the secret access key.
5. AWS generates a signature from the request data and the secret access key using the same algorithm used to calculate the signature in the request.
6. If the signatures match, the request is considered to be authentic. If the comparison fails, the request is discarded, and AWS returns an error response.

Note
If a request contains a *`Timestamp`* parameter, the signature calculated for the request expires 15 minutes after its value. If a request contains an *`Expires`* parameter, the signature expires at the time specified by the *`Expires`* parameter.

To calculate the request signature

1. Create the canonicalized query string that you need later in this procedure:

   a. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when Content-Type is application/x-www-form-urlencoded).

   b. URL encode the parameter name and values according to the following rules:

      i. Do not URL encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A–Z, a–z, 0–9, hyphen (-), underscore (_), period (.), and tilde (~).

      ii. Percent encode all other characters with %XY, where X and Y are hex characters 0–9 and uppercase A–F.

      iii. Percent encode extended UTF-8 characters in the form %XY%ZA....

      iv. Percent encode the space character as %20 (and not +, as common encoding schemes do).

   c. Separate the encoded parameter names from their encoded values with the equals sign (=) (ASCII character 61), even if the parameter value is empty.

   d. Separate the name-value pairs with an ampersand (&) (ASCII code 38).

2. Create the string to sign according to the following pseudo-grammar (the "\n" represents an ASCII newline).

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

The HTTPRequestURI component is the HTTP absolute path component of the URI up to, but not including, the query string. If the HTTPRequestURI is empty, use a forward slash (/).

3. Calculate an RFC 2104-compliant HMAC with the string you just created, your secret access key as the key, and SHA256 or SHA1 as the hash algorithm.

   For more information, go to RFC 2104.

4. Convert the resulting value to base 64.

5. Include the value as the value of the *Signature* parameter in the request.

For example, the following is an example request (line breaks added for clarity).

```
https://rds.amazonaws.com/
?Action=DescribeDBInstances
 &DBInstanceIdentifier=myinstance
 &Version=2010-01-01
 &Timestamp=2010-05-10T17%3A09%3A03.726Z
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &AWSAccessKeyId=<Your AWS Access Key ID>
```

For the preceding Query string, you calculate the HMAC signature over the following string.

```
GET\n
rds.amazonaws.com\n
AWSAccessKeyId=<Your AWS Access Key ID>
&Action=DescribeDBInstances
&DBInstanceIdentifier=myinstance
&Timestamp=2010-05-10T17%3A09%3A03.726Z
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Version=2009-10-16
```

The result is the following signed request.

```
https://rds.amazonaws.com/
?Action=DescribeDBInstances
 &DBInstanceIdentifier=myinstance
 &Version=2010-01-01
 &Timestamp=2010-05-10T17%3A09%3A03.726Z
 &Signature=<URLEncode(Base64Encode(Signature))>
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &AWSAccessKeyId=<Your AWS Access Key ID>
```

Using the SOAP API

Topics

- WSDL and Schema Definitions (p. 633)
- Programming Language Support (p. 633)
- Request Authentication (p. 634)
- Response Structure (p. 635)
- Web Services References (p. 636)

WSDL and Schema Definitions

You can access the Amazon Relational Database Service using the SOAP web services messaging protocol. This interface is described by a Web Services Description Language (WSDL) document, which defines the operations and security model for the particular service. The WSDL references an XML Schema document, which strictly defines the data types that might appear in SOAP requests and responses. For more information on WSDL and SOAP, see Web Services References (p. 636).

> **Note**
> Amazon RDS supports SOAP only through HTTPS.

All schemas have a version number. The version number appears in the URL of a schema file and in a schema's target namespace. This makes upgrading easy by differentiating requests based on the version number.

The current versions of the Amazon RDS WSDL are available at the following locations:

Region	WSDL Location
US East (N. Virginia) region	https://rds.us-east-1.amazonaws.com/doc/2014-10-31/AmazonRDSv7.wsdl
US West (N. California) region	https://rds.us-west-1.amazonaws.com/doc/2014-10-31/AmazonRDSv7.wsdl
US West (Oregon) region	https://rds.us-west-2.amazonaws.com/doc/2014-10-31/AmazonRDSv7.wsdl
EU (Ireland) region	https://rds.eu-west-1.amazonaws.com/doc/2014-10-31/AmazonRDSv7.wsdl
Asia Pacific (Singapore) region	https://rds.ap-southeast-1.amazonaws.com/doc/2014-10-31/AmazonRDSv7.wsdl
Asia Pacific (Tokyo) region	https://rds.ap-northeast-1.amazonaws.com/doc/2014-10-31/AmazonRDSv7.wsdl
South America (São Paulo) Region	https://rds.sa-east-1.amazonaws.com/doc/2014-10-31/AmazonRDSv7.wsdl
EU (Frankfurt) Region	https://rds.eu-central-1.amazonaws.com/doc/2014-10-31/AmazonRDSv7.wsdl

Programming Language Support

Since the SOAP requests and responses in Amazon RDS follow current standards, any programming language with the appropriate library support can be used. Languages known to have this support include C++, C#, Java, Perl, Python and Ruby.

Request Authentication

Amazon RDS complies with the current WS-Security standard, which requires you to hash and sign SOAP requests for integrity and non-repudiation. WS-Security defines profiles which are used to implement various levels of security. Secure SOAP messages use the BinarySecurityToken profile, consisting of an X.509 certificate with an RSA public key.

The following is the content of an insecure `DescribeDBInstances` operation:

```
<DescribeDBInstances>
    <MaxRecords>100<MaxRecords>
</DescribeDBInstances>
```

To secure the request, we add the `BinarySecurityToken` element.

The secure version of the request begins with the following:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xm
lns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLS
chema-instance">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd">
    <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="Timestamp-2">
      <wsu:Created>2009-10-28T18:41:59.597Z</wsu:Created>
      <wsu:Expires>2009-10-28T18:46:59.597Z</wsu:Expires>
    </wsu:Timestamp>
    <wsse:BinarySecurityToken
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
ity-utility-1.0.xsd"
    EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary"
     ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
token-profile-1.0#X509v3"
     wsu:Id="CertId-5992FC58FDECA60AF912567553195531"
     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-secext-1.0.xsd">
     ....many, many lines of base64 encoded X.509 certificate...
    </wsse:BinarySecurityToken>
   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="Signature-
1">
      <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
        <ds:Reference URI="#Timestamp-2">
          <ds:Transforms>
           <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

          </ds:Transforms>
         <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

          <ds:DigestValue>DLFQyK61qWoJiMyC9w34siRELAM=</ds:DigestValue>
```

```
        </ds:Reference>
        <ds:Reference URI="#id-3">
          <ds:Transforms>
           <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

          </ds:Transforms>
         <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

          <ds:DigestValue>gUnvvoUezxgt56eBl2kW/y5diMk=</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>OMoJJqqDnahRt/9H2n8obJolyVprpziAzlFRZ9KbdwX
JoD1Rl2sAikZ0IJW7/VS9q8GH4JDsT2v1
       UoUogKgRSWy3sU4943g1T0vhyigbUm4vNxE/qUKm
SIXx2ed/8buaF9oRiB8zYDu0/qRT+QQ73rdaoyn2YRNkSi2+6P2FHmE=
      </ds:SignatureValue>
      <ds:KeyInfo Id="KeyId-5992FC58FDECA60AF912567553195672">
        <wsse:SecurityTokenReference
         xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
         wsu:Id="STRId-5992FC58FDECA60AF912567553195703"
         xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
          <wsse:Reference URI="#CertId-5992FC58FDECA60AF912567553195531"
           ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"/>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </ds:Signature>
  </wsse:Security>
</soap:Header>
```

If you are matching this against requests generated by Amazon RDS supplied libraries, or those of another vendor, the following are the most important elements.

Elements

- **BinarySecurityToken—**Contains the X.509 certificate in base64 encoded PEM format
- **Signature—**Contains an XML digital signature created using the canonicalization, signature algorithm, and digest method
- **Timestamp—**Requests to Amazon RDS are valid within 5 minutes of this value to help prevent replay attacks

Response Structure

In response to a request, the Amazon RDS service returns an XML data structure that conforms to an XML schema defined as part of the Amazon RDS WSDL. The structure of an XML response is specific to the associated request.

The following is an example response:

```
<DescribeDBInstancesResponse xmlns="http://rds.amazonaws.com/admin/2009-10-16/">

  <DescribeDBInstancesResult>
    <DBInstances/>
  </DescribeDBInstancesResult>
  <ResponseMetadata>
    <RequestId>946cda70-c3f1-11de-807a-79c03c55f7d4</RequestId>
  </ResponseMetadata>
</DescribeDBInstancesResponse>
```

Web Services References

For more information about using web services, go to any of the following resources:

- Web Service Description Language (WSDL)
- WS-Security BinarySecurityToken Profile

Available Libraries

AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of SOAP and Query. These libraries provide basic functions (not included in the APIs), such as request authentication, request retries, and error handling so that it is easier to get started. Libraries and resources are available for the following languages:

- Java
- PHP
- Python
- Ruby
- Windows and .NET

For libraries and sample code in all languages, go to Sample Code & Libraries.

Troubleshooting Applications on Amazon RDS

Topics

- Retrieving Errors (p. 636)
- Troubleshooting Tips (p. 637)

Amazon Relational Database Service; provides specific and descriptive errors to help you troubleshoot problems while interacting with the Amazon RDS API.

Retrieving Errors

Typically, you want your application to check whether a request generated an error before you spend any time processing results. The easiest way to find out if an error occurred is to look for an `Error` node in the response from the Amazon RDS API.

XPath syntax provides a simple way to search for the presence of an *Error* node, as well as an easy way to retrieve the error code and message. The following code snippet uses Perl and the XML::XPath module to determine if an error occurred during a request. If an error occurred, the code prints the first error code and message in the response.

```
use XML::XPath;
                my $xp = XML::XPath->new(xml =>$response);
                if ( $xp->find("//Error") )
               {print "There was an error processing your request:\n", " Error
 code: ",
                $xp->findvalue("//Error[1]/Code"), "\n", " ",
                $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

Troubleshooting Tips

We recommend the following processes to diagnose and resolve problems with the Amazon Relational Database Service API.

- Verify that Amazon Relational Database Service is operating normally in the region you are targeting by visiting http://status.www.amazonaws.cn.
- Check the structure of your request
  Each Amazon Relational Database Service operation has a reference page in the *Amazon RDS API Reference*. Double-check that you are using parameters correctly. In order to give you ideas regarding what might be wrong, look at the sample requests or user scenarios to see if those examples are doing similar operations.
- Check the forum
  Amazon RDS has a development community forum where you can search for solutions to problems others have experienced along the way. To view the forum, go to
  https://forums.aws.csdn.net/

RDS REST API Reference

Standard API syntax cannot be used in certain scenarios. In these cases, a REST API is provided.

Amazon RDS provides the following REST APIs:

- DownloadCompleteDBLogFile (p. 637)

Related Topics

- RDS Query API Documentation

DownloadCompleteDBLogFile

Because a database log file can be arbitrarily large, the `DownloadCompleteDBLogFile` REST API is provided to enable streaming of the log file contents. This action can also be performed using the RDS console (go to Downloading a Database Log File (p. 609)), or the rds-download-db-logfile CLI command.

Description

Downloads the contents of the specified database log file.

Request Parameters

DBInstanceIdentifier
The customer-assigned name of the DB instance that contains the log file you want to download.

LogFileName
The name of the log file to be downloaded.

Syntax

```
GET /v13/downloadCompleteLogFile/DBInstanceIdentifier/LogFileName HTTP/1.1
Content-type: application/json
host: rds.region.amazonaws.com
```

Response Elements

The `DownloadCompleteDBLogFile` REST API returns the contents of the requested log file as a stream.

Errors

DBInstanceNotFound
`DBInstanceIdentifier` does not refer to an existing DB instance.

HTTP Status Code: 404

Examples

The following example downloads the log file named *log/ERROR.6* for the DB instance named *sample-sql* in the *us-west-2* region.

```
GET /v13/downloadCompleteLogFile/sample-sql/log/ERROR.6 HTTP/1.1
host: rds.us-west-2.amazonaws.com
X-Amz-Security-Token: AQoDYXdzEIH/////////wEa0AIXLhngC5zp9CyB1R6abwKrX
HVR5efnAVN3XvR7IwqKYalFSn6UyJuEFTft9nObglx4QJ+GXV9cpACkETq=
X-Amz-Date: 20140903T233749Z
X-Amz-Algorithm: AWS4-HMAC-SHA256
X-Amz-Credential: AKIADQKE4SARGYLE/20140903/us-west-2/rds/aws4_request
X-Amz-SignedHeaders: host
X-Amz-Content-SHA256: e3b0c44298fc1c229af
bf4c8996fb92427ae41e4649b934de495991b7852b855
X-Amz-Expires: 86400
X-Amz-Signature: 353a4f14b3f250142d9afc34f9f9948154d46ce7d4ec091d0cdabbcf8b40c558
```

For information about creating a version 4 signature, go to Signature Version 4 Signing Process.

Related Topics

- Downloading a Database Log File (p. 609)
- rds-download-db-logfile
- DownloadDBLogFilePortion
- RDS Query API Documentation

Resources for Amazon RDS

The following table lists related resources that you'll find useful as you work with Amazon RDS.

Resource	Description
Amazon Relational Database Service API Reference	The API reference contains a comprehensive description of all Amazon RDS Query APIs and data types.
Amazon Relational Database Service Command Line Reference	The Command Line Tools Reference contains a comprehensive description of all the command line tools and their options.
Amazon RDS Technical FAQ	The FAQ covers the top questions developers have asked about this product.
Release notes	The release notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
AWS Developer Resource Center	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
AWS Management Console	The AWS Management Console allows you to perform most of the functions of Amazon RDS without programming.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
AWS Support Center	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and Premium Support.
Amazon RDS product information	The primary web page for information about Amazon RDS.
Contact Us	A central contact point for inquiries concerning AWS billing, account, events, abuse etc.
Conditions of Use	Detailed information about the copyright and trademark usage at Amazon.com and other topics.

Document History

The following table describes the important changes to the documentation since the last release of the *Amazon Relational Database Service User Guide*.

- **API version:** 2014-10-31
- **Latest documentation update:** July 30, 2015

Change	Description	Date Changed
New feature	Updated to support PostgreSQL versions 9.4.4 and 9.3.9. For more information, see Supported PostgreSQL Database Versions (p. 357).	July 30, 2015
New feature	Microsoft SQL Server Enterprise Edition is now available with the License Included service model. For more information, see License Included (p. 306).	July 29, 2015
New feature	Amazon RDS for Aurora has officially released. The Amazon Aurora DB engine supports multiple DB instances in a DB cluster. For detailed information, see Aurora on Amazon RDS (p. 392).	July 27, 2015
New feature	Updated to support copying tags to DB snapshots.	July 20, 2015
New feature	Updated to support Oracle 12c database version "12.1.0.2", including the In-Memory option, Oracle 11g April PSU patches, and improved integration with AWS CloudHSM.	July 20, 2015
New feature	Updated to support increases in storage size for all DB engines and an increase in Provisioned IOPS for SQL Server.	June 18, 2015
New feature	Updated options for reserved DB instances.	June 15, 2015
New feature	Updated to support Oracle version 12c.	April 2, 2015
New feature	Updated to support PostgreSQL versions 9.3.6 and 9.4.1.	March 18, 2015
New feature	Updated to support using Amazon CloudHSM with Oracle DB instances using TDE.	January 8, 2015

Change	Description	Date Changed
New feature	Updated to support encrypting data at rest and new API version 2014-10-31.	January 6, 2015
New feature	Updated to support Oracle versions 11.2.0.3.v2 and 11.2.0.4.v3 that include the PSU released in October 2014.	November 20, 2014
New feature	Updated to include the new Amazon DB engine: Aurora. The Amazon Aurora DB engine supports multiple DB instances in a DB cluster. Amazon Aurora is currently in preview release and is subject to change. For detailed information, see Aurora on Amazon RDS (p. 392).	November 12, 2014
New feature	Updated to support PostgreSQL Read Replicas.	November 10, 2014
New features	Updated to support Oracle 11.2.0.4v2.	October 16, 2014
New API and features	Updated to support the GP2 storage type and new API version 2014-09-01. Updated to support the ability to copy an existing option or parameter group to create a new option or parameter group.	October 7, 2014
New feature	Updated to support InnoDB Cache Warming for DB instances running MySQL version 5.6.19 and later.	September 3, 2014
New feature	Updated to support SSL certificate verification when connecting to MySQL version 5.6, SQL Server, and PostgreSQL database engines.	August 5, 2014
New feature	Updated to support the db.t2 burst-capable DB instance classes.	August 4, 2014
New feature	Updated to support the db.r3 memory-optimized DB instance classes for use with the MySQL (version 5.6), SQL Server, and PostgreSQL database engines.	May 28, 2014
New feature	Updated to support SQL Server Multi-AZ deployments using SQL Server Mirroring.	May 19, 2014
New feature	Updated to support upgrades from MySQL version 5.5 to version 5.6.	April 23, 2014
New feature	Updated to support Oracle 11.2.0.4.	April 23, 2014
New feature	Updated to support Oracle GoldenGate.	April 3, 2014
New feature	Updated to support the M3 DB instance classes.	February 20, 2014
New feature	Updated to support the Oracle Timezone option.	January 13, 2014
New feature	Updated to support Oracle 11.2.0.3.	December 16, 2013
New feature	Updated to support replication between Amazon RDS MySQL DB instances in different regions.	November 26, 2013
New feature	Updated to support the PostgreSQL DB engine.	November 14, 2013
New feature	Updated to support SQL Server transparent data encryption (TDE).	November 7, 2013

Change	Description	Date Changed
New API and new feature	Updated to support cross region DB snapshot copies; new API version, 2013-09-09.	October 31, 2013
New features	Updated to support Oracle Statspack.	September 26, 2013
New features	Updated to support using replication to import or export data between instances of MySQL running in Amazon RDS and instances of MySQL running on-premises or on Amazon EC2.	September 5, 2013
New features	Updated to support the db.cr1.8xlarge DB instance class for MySQL 5.6.	September 4, 2013
New feature	Updated to support replication of Read Replicas.	August 28, 2013
New feature	Updated to support parallel Read Replica creation.	July 22, 2013
New feature	Updated to support fine-grained permissions and tagging for all Amazon RDS resources.	July 8, 2013
New feature	Updated to support MySQL 5.6 for new instances, including support for the MySQL 5.6 memcached interface and binary log access.	July 1, 2013
New feature	Updated to support major version upgrades from MySQL 5.1 to MySQL 5.5.	June 20, 2013
New feature	Updated DB parameter groups to allow expressions for parameter values.	June 20, 2013
New API and new feature	Updated to support Read Replica status; new API version, 2013-05-15.	May 23, 2013
New features	Updated to support Oracle Advanced Security features for native network encryption and Oracle Transparent Data Encryption.	April 18, 2013
New features	Updated to support major version upgrades for SQL Server and additional functionality for Provisioned IOPS.	March 13, 2013
New feature	Updated to support VPC By Default for RDS.	March 11, 2013
New API and feature	Updated to support log access; new API version 2013-02-12	March 4, 2013
New feature	Updated to support RDS event notification subscriptions.	February 4, 2013
New API and feature	Updated to support DB instance renaming and the migration of DB security group members in a VPC to a VPC security group.	January 14, 2013
New feature	Updated for AWS GovCloud (US) support.	December 17, 2012
New feature	Updated to support m1.medium and m1.xlarge DB Instance classes.	November 6, 2012
New feature	Updated to support Read Replica promotion.	October 11, 2012
New feature	Updated to support SSL in Microsoft SQL Server DB Instances.	October 10, 2012

Change	Description	Date Changed
New feature	Updated to support Oracle micro DB Instances.	September 27, 2012
New feature	Updated to support SQL Server 2012.	September 26, 2012
New API and feature	Updated to support provisioned IOPs. API version 2012-09-17.	September 25, 2012
New features	Updated for SQL Server support for DB Instances in VPC and Oracle support for Data Pump.	September 13, 2012
New feature	Updated for support for SQL Server Agent.	August 22, 2012
New feature	Updated for support for tagging of DB Instances.	August 21, 2012
New features	Updated for support for Oracle APEX and XML DB, Oracle time zones, and Oracle DB Instances in a VPC.	August 16, 2012
New features	Updated for support for SQL Server Database Engine Tuning Advisor and Oracle DB Instances in VPC.	July 18, 2012
New feature	Updated for support for MySQL db.t1.micro DB Instances.	June 11, 2012
New feature	Updated for support for option groups and first option, Oracle Enterprise Manager Database Control.	May 29, 2012
New feature	Updated for support for Read Replicas in Amazon Virtual Private Cloud.	May 17, 2012
New feature	Updated for Microsoft SQL Server support.	May 8, 2012
New features	Updated for support for forced failover, Multi-AZ deployment of Oracle DB Instances, and nondefault character sets for Oracle DB Instances.	May 2, 2012
New feature	Updated for Amazon Virtual Private Cloud (VPC) Support.	February 13, 2012
Updated content	Updated for new Reserved Instance types.	December 19, 2011
New feature	Updated for Oracle engine support.	May 23, 2011
Updated content	Console updates.	May 13, 2011
Updated content	Edited content for shortened backup and maintenance windows.	February 28, 2011
New feature	Added support for MySQL 5.5.	January 31, 2011
New feature	Added support for Read Replicas.	October 4, 2010
New feature	Added support for AWS Identity and Access Management (IAM).	September 2, 2010
New feature	Added DB Engine Version Management.	August 16, 2010
New feature	Added Reserved DB Instances.	August 16, 2010
New Feature	Amazon RDS now supports SSL connections to your DB Instances.	June 28, 2010
New Guide	This is the first release of *Amazon Relational Database Service User Guide*.	June 7, 2010